# ΔΣ ADC Linearity Testing Technology and Floating-point Arithmetic Algorithms with Taylor-series Expansion

**JIANGLIN WEI**


**PhD Dissertation**

**DIVISION OF ELECTRONICS & INFORMATICS**
**GRADUATE SCHOOL OF SCIENCE & TECHNOLOGY**
GUNMA UNIVERSITY

JAPAN

March 2022

# SΔΣ ADC Linearity Testing Technology and Floating-point Arithmetic Algorithms with Taylor-series Expansion

**DISSERTATION**

*Submitted by*

**JIANGLIN WEI**

*In partial fulfillment of the requirements for the award of the Degree of*

**DOCTOR OF PHILOSOPHY**
**IN**
**ELECTRONICS & INFORMATICS ENGINEERING**

Under the guidance of

**PROFESSOR HARUO KOBAYASHI, Ph. D. Eng.**

**DIVISION OF ELECTRONICS & INFORMATICS**
**GRADUATE SCHOOL OF SCIENCE & TECHNOLOGY**
GUNMA UNIVERSITY
JAPAN

March 2022

# Acknowledgement

As I approach the end of my Ph.D. research, I would like to thank all those who have helped me in my life and studies at this important time in my life.

Firstly, I sincerely thank Prof. Haruo Kobayashi who directly guided me to implement this dissertation. Prof. Kobayashi has been a great example of academic and career achievement, and has been a guiding example to us all. In life, Prof. Kobayashi is a humble and caring person who has provided us with excellent conditions for research; in academia, he is innovative and capable, and has always given us much help and guidance in our daily research, which has benefited us greatly. He also encourages students to participate in various international conferences so that they can learn about the current direction of science and technology, the most advanced technology, as well as exchange and study with scholars from different countries, so that our self-learning ability can be improved rapidly.

I would like to express sincere gratitude to my review committee members, Prof. Yasushi Yuminaka, Prof. Shugang Wei, Prof. Yusaku Fujii, and Prof. Kuniyuki Motojima for their careful guidance and very valuable suggestions on the completion of my Ph.D. dissertation.

In addition, I would like to express my appreciation to Assistant Prof. Anna Kuwana for her valuable discussion, advice, and support during my present in Kobayashi Lab. Moreover, we would also like to thank ROHM Semiconductor Co., Ltd. for supporting this work, and especially Mr. Keno Sato, Mr. Takashi Ishida, Mr. Tomoyuki Okamoto, and Mr. Tamotsu Ichikawa for many useful comments. I thank them very much.

Furthermore, I would like to thank Kobayashi Lab research team for always caring, supporting and creating favorable conditions for me throughout the implementation of dissertation.

Finally, sincerely thanks to my parents. Thanks for their patience, support and endless love that give me courage. Special thanks also to my

brother for taking care of my family while I was away from home.

# Declaration

I hereby undertake: This dissertation is my own research work.

There is no reproduction of the results in any previously published documents or articles. Dissertation is done under the scientific guidance of Prof. HARUO KOBAYASHI. The data, images, and research results presented in the dissertation are completely honest. Dissertation has references and uses materials posted on conferences, magazines, articles, websites, textbooks. All materials used in the dissertation are mentioned in detail in the references section.

I take full responsibility for the above guarantees.

**Signature:**

**Name: JIANGLIN WEI**

**Student No.:** T192D601

**Date:**

# Abstract

This dissertation consists of two main parts; the first part consists of a high precision ΔΣ digital-to-analog converter (DAC) technique and a fast test technique for integral nonlinearity (INL) of a ΔΣ analog-to-digital converter (ADC). For ΔΣ DA conversion a limit cycle is generated, which can easily cause the accuracy degradation during DA conversion. To address this problem, we propose a technique to add a random signal. In the general DA converter model, one port of the quantizer is grounded, for our proposed method is to change the grounded port of the quantizer to our proposed random signal, and use the random signal added to the grounded side of the quantizer to suppress the effect of the resulting limit cycle, thus improving the DA conversion accuracy. We all also compare the effects of adding random signals to the quantizer in different ranges to obtain the best range of input random signals. Another one is high-speed INL testing technology of the ΔΣ ADC with our proposed FFT method. The ΔΣ ADC is now widely used in sensor interface circuits of Internet of Things (IoTs) systems. It is high-precision but its sampling speed is very slow so that its direct INL testing time takes very long time and then its INL testing is usually omitted at mass production shipping. However, due to the demands for low-cost yet highly-reliable IoT systems, its short time testing is now needed. We consider its INL testing by separating its analog and digital parts: ΔΣ AD modulator and digital filter. The digital filter can be tested with the scan-path method. We consider a polynomial model of the ΔΣ AD modulator input-output (I/O) characteristics and estimate its coefficient values from the fundamental and harmonics power by applying a cosine input and obtaining the power spectrum of the modulator 1-bit output stream with FFT.

The second part describes the study of arithmetic algorithms for floating-point numbers using Taylor-series development. In high-speed digital signal processing, the need for high accuracy, low latency, and low power are often the problems we need to overcome. For these problems, we have developed floating-point digital arithmetic algorithms to deal with division, inverse square root, logarithm and exponential calculations, using Taylor-series expansion. Three methods are proposed for fast calculation with high efficiency: (i) mantissa region uniform division, (ii) mantissa region non-uniform division and (iii) mantissa region conversion. We have clarified the Taylor-series expansion calculation algorithm trade-offs among accuracy, numbers of multiplications/additions/ subtractions and LUT sizes. The simplest and most efficient algorithm in different arithmetic using the three methods is proposed in this paper.

# Contents

# List of Figures

## PART 1

## PART 2

# List of Tables

## PART 1



## PAER 2

# PART 1

# Chapter 1

# INTRODUCTION

## 1.1 Research Background and Motivation

With the development of digital signal processing and digital computing technology, we are now enjoying life in the "digital" world more and more [1, 2]. Compared to analog circuits, using digital circuits is less affected by noise and more reliable. The advantage is that it is easier to integrate on a chip to realize complex functions. However, the signals touched in the real world are analog signals such as sounds and images. Therefore, conversion between analog signals and digital circuit signals is important. It is necessary to be able to easily convert an analog signal into a digital signal as an interface between them. The circuit that realizes this function is an A/D Converter (Analog-to-Digital Converter, ADC) [3].

ADCs are important modules in the configuration of electronic systems and often have a significant impact on the overall performance of the system. With the development of ultra-deep submicron CMOS processes, the degree of digital circuit integration increases [4, 5]. The price is getting higher and the function to realize it is complicated while the high-speed processing of signals and the demand for accuracy are getting higher. However, the development of ADC design is relatively slow, and the development of analog design software is immature. The development of analog interface circuits lags behind the development of digital circuits. Especially in digital television video systems and digital communication systems, their performance (e. g., speed and accuracy) often limits the improvement of overall system performance [6,7].

Since the introduction of ADC, we have experienced the development of data conversion of discrete semiconductors and integrated circuits, and the development strategy of high-speed and high-precision ADC is to increase the degree of integration as much as possible and solve the product for the final user, assuming that the performance is not affected. It has gone through the process of semiconductor, integrated circuit data conversion to provide a solution. Nowadays, the demand for ADC is increasing significantly, and the performance index is wider in order to adapt to the requirements applied to various applications. It is required to be covered. The main fields of application of ADCs are constantly expanding, and they are widely applied in fields such as sensor, DSP, multimedia, communication, and measurement [8,9]. ADCs meet the different demands of various fields. On the other hand, at the design stage, not only the process and circuit configuration of the ADC itself, but also the signal modulation is supported, and the peripheral circuit design of the ADC such as an analog circuit such as an analog filter is also performed. These should be taken into consideration.

## 1.2 Organization

In this part, the research background and motivation are introduced and dissertation organization in Chapter 1. In Chapter 2, the AD conversion principle, the evaluation criteria for ADCs, and the structure and application scenarios of different ADCs are introduced. Chapter 3 proposes that random signals can improve the ΔΣ AD conversion accuracy. Chapter 4 presents the high-speed INL testing method for ΔΣ ADC. Chapter 5 summarizes this part and future work.

# References


[1] L. Wanhammar, DSP Integrated Circuits. New York: Academic, 1999.

[2] J. Eyre and J. Bier, "The Evolution of DSP Processors", IEEE Signal Processing Magazine, vol. 17, no. 2, pp. 43-51 (Mar. 2000).

[3] B. Murmann, "A/D Converter Trends: Power Dissipation, Scaling and Digitally Assisted Architectures", IEEE Custom Integrated Circuits Conference, San Jose, CA, USA (Sept. 2008).

[4] A. -. Annema, B. Nauta, R. van Langevelde, H. Tuinhout, "Analog Circuits in Ultra-Deep-Submicron CMOS", IEEE Journal of Solid-State Circuits, vol. 40, no. 1, pp. 132-143 (Jan. 2005).

[5] M. U. Khan, A. Arshad, S. Ume, R. Bukhari, A. Samer, and F. Tariq, "Nanoscale CMOS Scaling Trends, Challenges and Their Solutions." ICAME21, International Conference on Advances in Mechanical Engineering, Pakistan (Aug. 2021).

[6] T. Mao, Q. Wang and Z. Wang, "Spatial Modulation for Terahertz Communication Systems with Hardware Impairments," IEEE Transactions on Vehicular Technology, vol. 69, no. 4, pp. 4553-4557 (Apr. 2020).

[7] G. A. Ajenikoko, O. S. Olaniyan, A. V. Taye, et. al. "Design and Implementation of an Improved Digital Video Surveillance System", Computer Engineering and Intelligent Systems, vol. 11, no. 2, pp. 60-67 (Apr. 2020).

[8] P. A. H. Vardhini, M. L. Makkena. "Design and Comparative Analysis of On-Chip ΔΣ ADC for Signal Processing Applications", International Journal of Speech Technology, vol. 24, no. 2, pp. 401-407 (Jan. 2021).

[9] M. Gharaei Jomehei, S. Sheikhaei, M. Saberi. "A Low-Power, Low-Data-Rate Efficient ADC With Hybrid Exponential-Linear Transfer Curve For Bio-Potential Recording Systems", International Journal of Circuit Theory and Applications, vol. 48, no. 4, pp. 449-471 (Feb. 2020).

# Chapter 2

# A/D CONVERTER MAIN STRUCTURE AND PERFORMANCE INDEX

In today's society, digital signals are developing rapidly, and digital signal processing (DSP) algorithms are becoming more and more important and relevant to our lives. Real-life signals generally exist in the form of analog signals, which are continuous in the time domain and exhibit values that can be arbitrary at any moment in time. For example, temperature, sound, light, and you name it. However, digital signals are discrete in time and discrete in amplitude, manifesting themselves with specific values at specific moments.

Compared with analog circuits, digital circuits have the advantages of easy automatic control, easy storage, and high immunity to interference. Therefore, many analog signals are converted to digital signals for processing to achieve high-speed and high-precision signal processing and transmission. As shown in Fig. 2.1, first the A/D converter converts an analog signal into a digital signal for transmission to the DSP for processing, and then the result of the processing is converted to an analog signal by the digital-to-analog converter (DAC). The conversion principle of ADC usually needs to go through four processes: sampling, holding, quantization and coding, which are represented in Fig. 2.2.

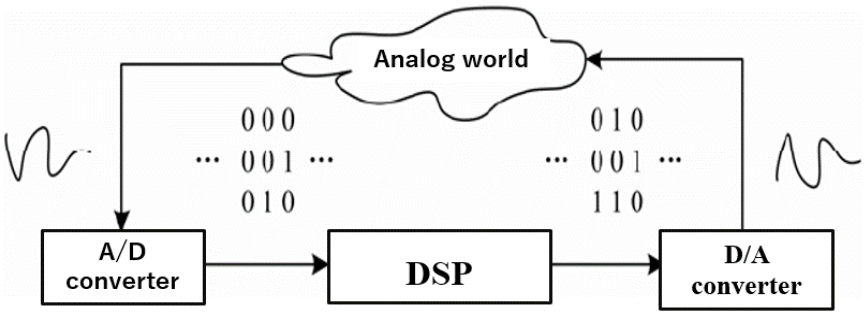



Fig. 2.1 Data conversion process.

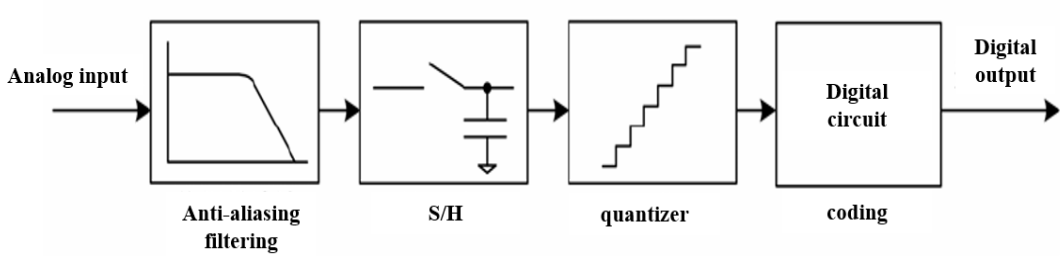
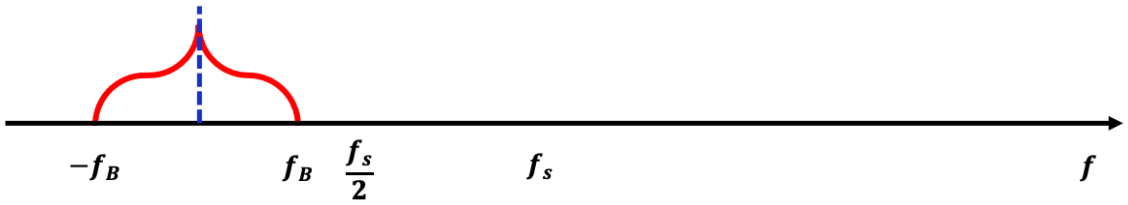


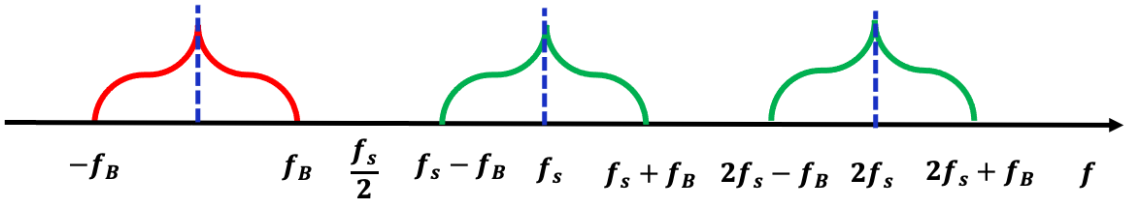

Fig. 2.2 Basic structure of A/D converter.

## 2.1 Sampling

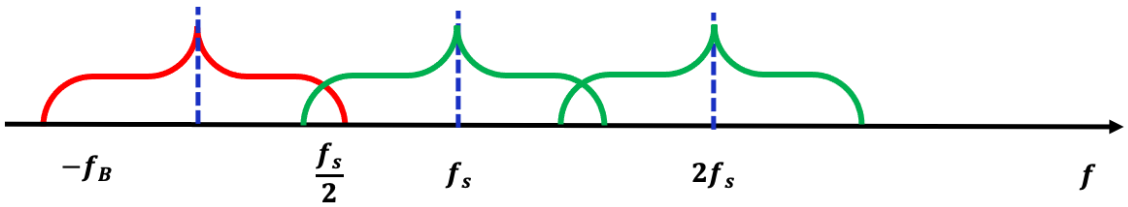
To ensure that the ADC works properly, it is necessary to understand the frequency response characteristics of the ADC, as shown in Fig. 2.3. Suppose that the frequency response of the analog input signal is shown in Fig. 2.3 (a), and the maximum frequency of the analog input signal is $f_B$, then when the analog input signal is sampled at a frequency of $f_S$, its frequency response is shown in Fig. 2.3 (b), and the spectrum of the input signal is repeated at the sampling frequency as well as at each of its harmonics. If the signal bandwidth $f_B$ is larger than $0.5f_S$, the spectrum will be mixed, as shown in Fig. 2.3 (c), and there it is impossible to recover the original signal; so it must be ensured that the input signal bandwidth is less than $0.5f_S$.



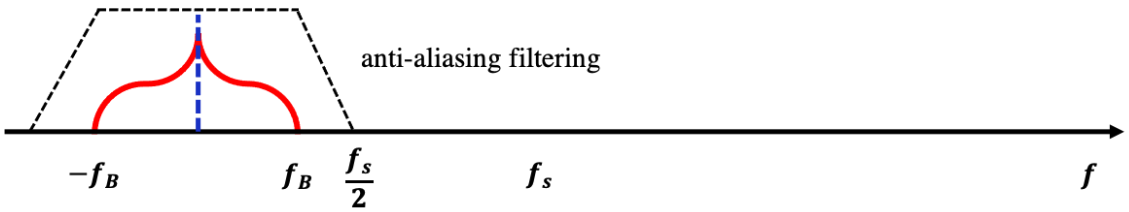

(a) Continuous time frequency response of analog input signal




(b) Equivalent frequency response of data sampling




(c) Mixing is caused when $f_B > 0.5f_S$




(d) Use of an anti-aliasing filter to avoid aliasing

Fig. 2.3 Frequency response of the ADC

As described above is the basic law of sampling, called Nyquist's sampling theorem [1], which describes that the sampling frequency must be more than twice the bandwidth of the input signal to guarantee the recovery of the original signal from the sampled signal, that is what the sampling frequency $f_S$ must satisfy:

$$2f_B < f_S \tag{2-1}$$

Where $f_B$ is the high frequency component of the input signal. In order to maximize the spectrum of the input signal, it is necessary to make $f_B$ as close as possible to $0.5f_S$, so that a pre-filter with very steep variations is required to eliminate the signal outside the $0.5f_S$ frequency, the specific spectrum is shown in Fig. 2.3(d), and it is very difficult and complicated to implement such a filter. The sampling frequency is usually taken as:

$$f_S \cong (2\sim3)\ f_B \tag{2-2}$$

The Nyquist sampling theorem shows the correlation between sampling frequency and signal frequency, and ADCs that work in this way are called Nyquist ADCs. There is also another ADC with a sampling frequency much higher than two times the signal bandwidth, called an oversampling ADC.

## 2.2 Quantization noise

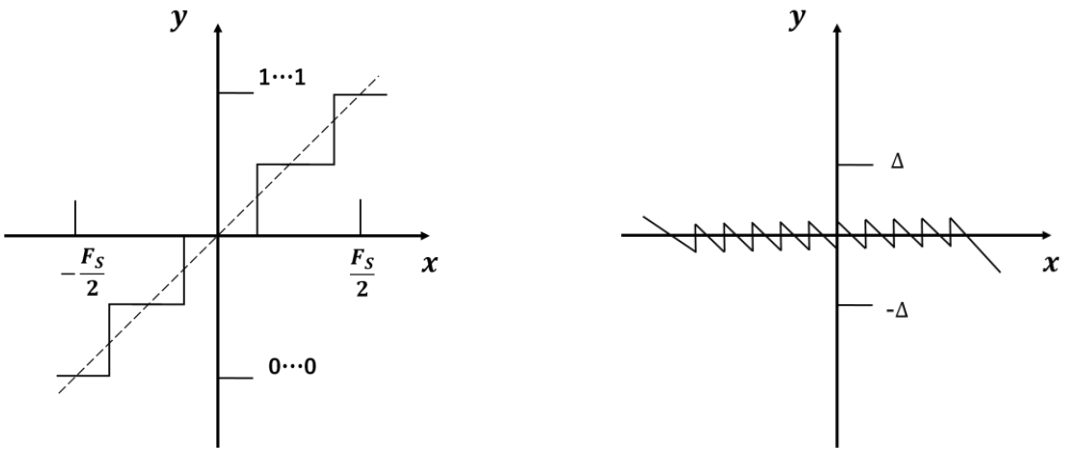
Fig. 2.4 (a) shows the input-output curve of the quantizer, which converts the input sample signal $x(n)$ into a discrete digital signal of amplitude $00\cdots0$ to $11\cdots1$, depending on the input size. Use up to N-bit digital signal. It means that the minimum input is $-F_S/2$ $(00\cdots0)$ and the maximum input is $+F_S/2$ $(11\cdots1)$. Here, $F_S$ is the full-scale input of the quantizer. The number of N-bit in the digital signal, represents the resolution of the quantizer. The number of N-bit of the digital signal represents the resolution of the quantizer. That is, the minimum step size that the quantizer can decompose is $\Delta= F_S/2^N$. The quantization error is the difference between the input and the output, and the relationship between the input signal and the quantization is shown in Fig. 2.4(b), where the quantization

error is limited to the range $[-F_S/2 \sim +F_S/2]]$ when the input signal size is limited to the range $[-\Delta/2 \sim +\Delta/2]$. Under certain conditions, such as random fluctuations of the input signal or limited input range, the quantization error can be regarded as uncorrelated random white noise, evenly distributed in $[-\Delta/2 \sim +\Delta/2]$. The Probability Density Function (PDF) is shown in Fig. 2.5(a). The total quantization noise energy can be calculated as follows:

$$\sigma_q^2 = \int_{-\infty}^{\infty} PDF(q)dq = \frac{1}{\Delta}\int_{-\Delta/2}^{\Delta/2} q^2 dq = \frac{\Delta^2}{12} \tag{2-3}$$

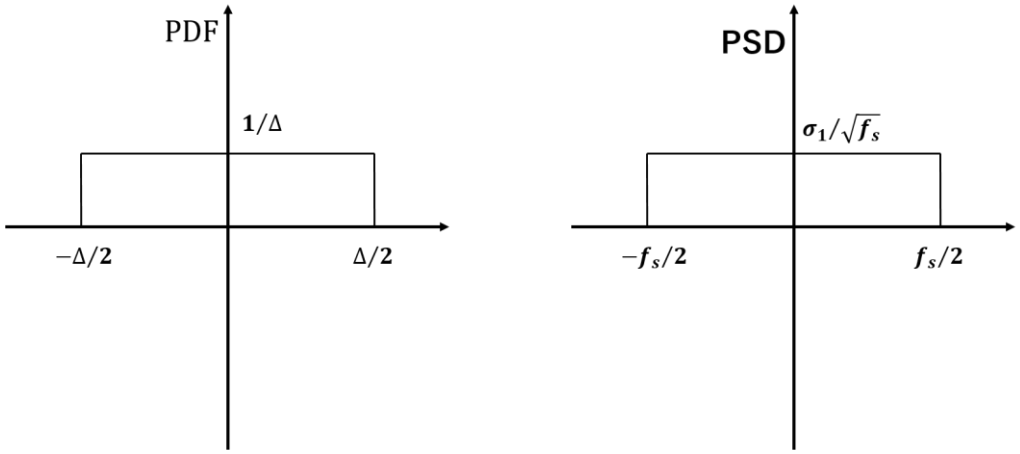
Its Power Spectral Density (PSD) is white noise evenly distributed in frequency domain over $[-f_s/2 \ , \ f_s/2]$ as shown in Fig. 2.5(b), its amplitude is $\Delta/\sqrt{12f_s}$, and the total noise energy is also obtained by integrating the PSD from $[-f_s/2 \ , \ f_s/2]$.

$$\sigma_q^2 = \int_{-f_s/2}^{f_s/2} \left(\frac{\Delta}{\sqrt{12f_s}}\right)^2 df = \frac{\Delta^2}{12} \tag{2-4}$$




(a) Quantized input/output. (b) Quantized noise.

Fig. 2.4 Quantizer.




(a) Probability Density Function. (b) Power Spectral Density.

Fig. 2.5 Quantization noise.

## 2.3 Performance indicators

### 2.3.1 Dynamic characteristics of ADC

The dynamic characteristics of an ADC are generally obtained by Fast Fourier Transform (FFT) analysis, which is related to the input frequency, input signal amplitude, and sampling frequency, and they are measured by Signal-to-Noise Ratio (SNR), Signal-to-Noise and Distortion Ratio (SNDR), Effective Number of Bits (ENOB), Spurious-Free Dynamic Range (SFDR), and Total Harmonic Distortion (THD), etc.

(1) SNR: Signal-to-Noise

It refers to the ratio of the energy of the signal to the energy of the noise in the signal band (usually measured with a sinusoidal signal input). Assuming that the input is a $F_S$ (Full Scale) sinusoidal signal (amplitude: $\mathrm{A} = F_S / 2$) and that the noise contains only quantization errors, the maximum signal-to-noise ratio is given by the following formula:

$$\mathrm{SNR} = \frac{1/2A^2}{\sigma_q^2} = 3 \times 2^{2N-1} \tag{2-5}$$

Expressed in decibels (dB), it is expressed as follows:

$$\mathrm{SNR[dB]} = 10log_{10}(SNR) = 6.02N + 1.76 \tag{2-6}$$

(2) SNDR: Signal-to-Noise and Distortion Ration

The signal distortion ratio refers to the ratio of the output signal power to the sum of all noise and harmonic power in the band, which is simply the ratio of the output signal power to the output non-signal power. It can be expressed as follows:

$$\mathrm{SNDR} = \frac{P_{signal}}{P_{noise} + P_{distortion}} \tag{2-7}$$

Where, $P_{signal}$ is the signal power, $P_{noise}$ is the noise power and $P_{distortion}$ is the total harmonic power.

(3) ENOB: Effective Number of Bits

The number of valid bits refers to the effective number of bits corresponding to the SNDR obtained by the ADC output at full scale input signal, with the following conversion relationship:

$$\mathrm{ENOB} = \frac{SNDR[dB] - 1.76}{6.02} \tag{2-8}$$

(4) SFDR: Spurious-Free Dynamic Range

The Spurious-Free Dynamic Range (SFDR) is defined as the ratio of the energy of the fundamental component to the maximum spurious component in the output signal of the ADC, which reflects the maximum

interference to the output signal of the ADC in a certain frequency band. For a certain input frequency, the input amplitude and the sampling frequency, the SFDR can be expressed as:

$$\mathrm{SFDR} = 20\log_{10}\frac{\mathrm{A_S}}{\mathrm{A_{HD(max)}}}\,[\mathrm{dB}] \tag{2-9}$$

Where, $\mathrm{A_S}$ denotes the effective value of the fundamental wave component of the ADC output signal under a certain sinusoidal input condition, $\mathrm{A_{HD(max)}}$ denotes the effective value of the maximum spurious signal output of the ADC.

(5) THD: Total Harmonic Distortion

Total Harmonic Distortion (THD) is defined as the ratio of the harmonic component of the ADC output signal to the energy of the fundamental signal and it can be expressed as:

$$\mathrm{THD} = 20\log_{10}\left(\frac{\sqrt{A_{HD2}^2 + A_{HD2}^2 + \cdots + A_{HD2}^2}}{A_s}\right) \tag{2-10}$$

Where, $A_s$ denotes the effective value of the fundamental component of the ADC output signal, and $A_{HDk}$ denotes the effective value of the $k^{\text{th}}$ harmonic in the output signal, $k = 2, 3, \cdots, m$. Since the energy of the higher harmonic components is usually smaller, $m = 6$ is generally taken.

### 2.3.2 Static characteristics of ADC

The static performance index of the ADC is the performance index that can be measured at low frequency input or even fixed voltage input, which includes Differential Non-linearity (DNL), Integral Non-linearity (INL),

Missing Code, Monotonicity, Offset Error, Gain Error, and so on. These parameters reflect the deviation of the actual quantization curve of the ADC from the ideal curve. The ideal quantization curve of a 3-bit ADC is shown in Fig. 2.4.

One very important parameter for the ADC is the Least-Significant Bit (LSB). LSB is the lowest weighted bit of the ADC digital output code, which usually corresponds to the analog value it represents. Any ADC is limited in its ability to recognize an analog input, and the metric that characterizes this ability is the resolution (also called precision), which can be expressed as a percentage of the ADC's full scale, or as the number of N bits, when the ADC has $2^N$ possible digital output codes. The LSB, resolution and ADC Full Scale Input Range (FSR) are related by the following equation:

$$1\text{LSB} = \frac{FSR}{2^N - 1} \cong \frac{FSR}{2^N} \tag{2-11}$$

Where the 1 in the denominator is not required for differential ADCs.

Corresponding to the LSB, the Most Significant Bit (MSB) refers to the highest bit of the ADC digital output code, which is generally half of the FSR.

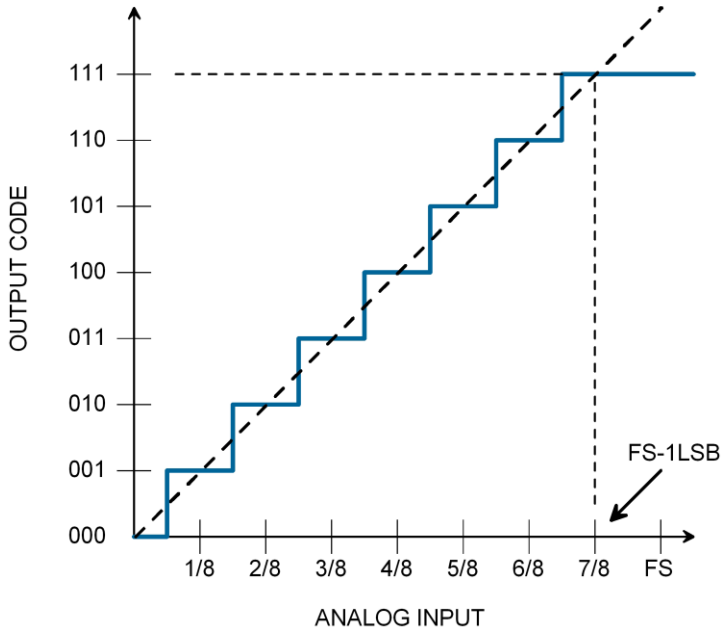



Fig. 2.4 Ideal 3-bit A/D converter quantifier curve.

(1) DNL: Differential Non-linearity

The ideal quantization curve digital code conversion width (the difference between two adjacent conversion levels) of the ADC is 1 LSB. DNL reflects the deviation of the actual quantization curve digital code conversion width of the ADC from the ideal value of 1 LSB, as shown in Fig. 2.5.

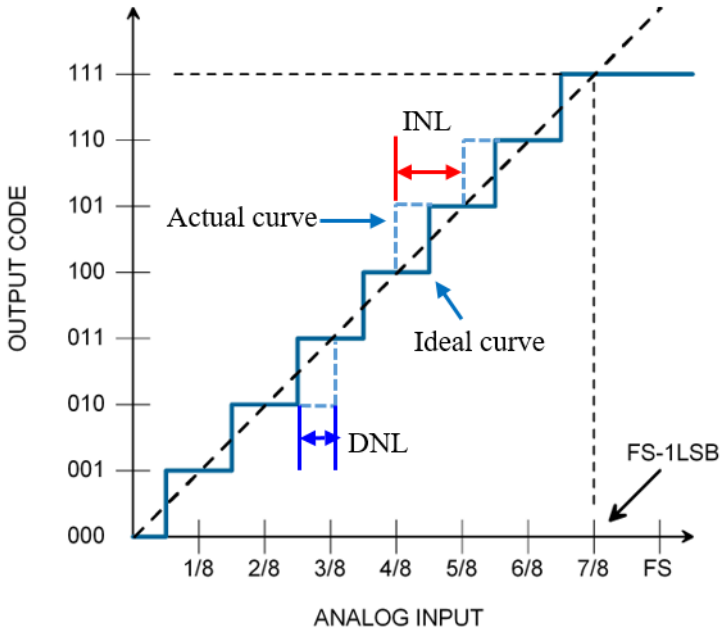



Fig. 2.5 DNL and INL.

(2) INL: Integral Non-linearity

All the conversion levels of the ideal quantization curve of the ADC are located in a straight line, while the actual quantization curve is not. INL reflects the deviation between the actual conversion level of the ADC and the ideal conversion level. An example is shown in Fig. 2.5. For a certain digital output code $M$, its corresponding INL and DNL have the following relationship:

$$\mathrm{INL}(M) = \sum_{i=0}^{M} \mathrm{DNL}(M) \tag{2-12}$$

(3) Missing code

If the ADC always has a certain (or some) digital code that cannot be generated regardless of the input voltage, it is said that the ADC has missing codes. If there is a missing code, it means that the ADC appears $\mathrm{DNL} = -1$

as shown in Fig. 2.6.

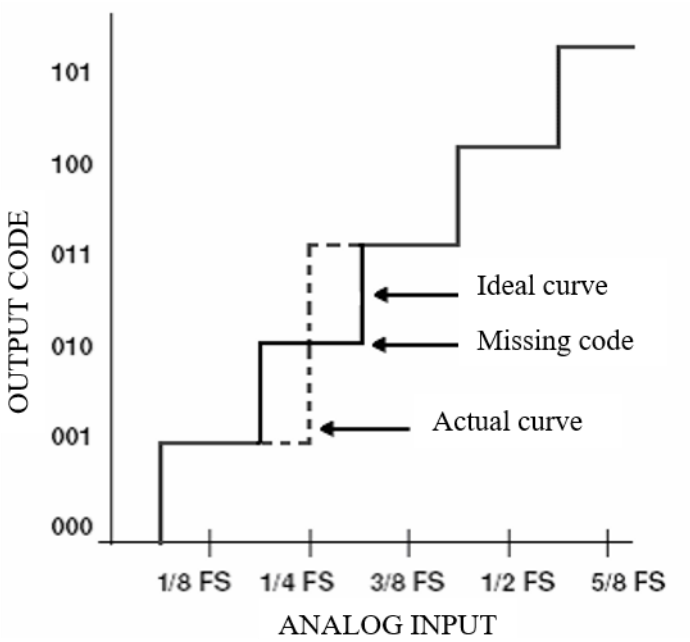



Fig. 2.6 Missing code.

(4) Monotonicity

Normally, each vertical step of the ADC quantization curve is positive (one digital code pitch), which means that the magnitude of the digital output code varies in the same direction as the amplitude of the input signal. If the quantization curve of the ADC has a negative jump in the vertical direction, the ADC has non-monotonic characteristics. This is shown in Fig. 2.7. Non-monotonicity worsens quantization noise and, if present in a closed-loop system, affects system stability [2-4].

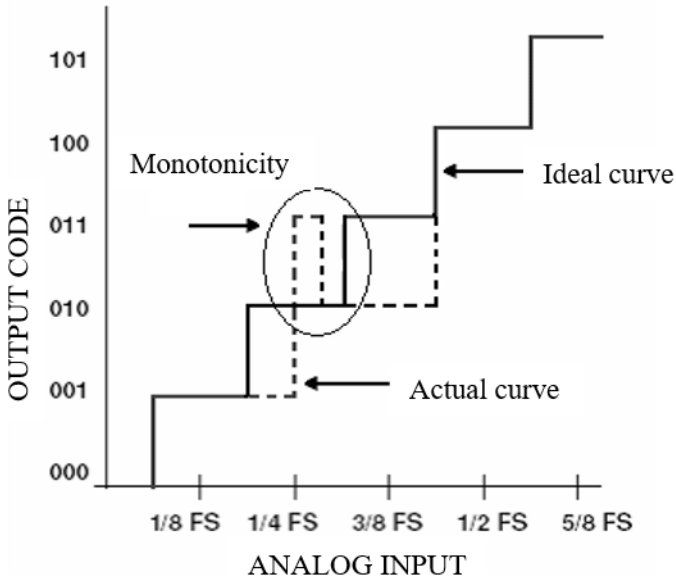



Fig. 2.7 Monotonicity.

(5) Offset error

The first conversion level of the ideal quantization curve of the ADC (the input analog voltage when the digital output code jumps from $00\cdots0$ to $00\cdots1$) is 0.5 LSB, and the offset error of the ADC is defined as the deviation of the first conversion level of the actual quantization curve from 0.5 LSB. The ADC shown in Fig. 2.8 has an offset error of 1.5 LSB.

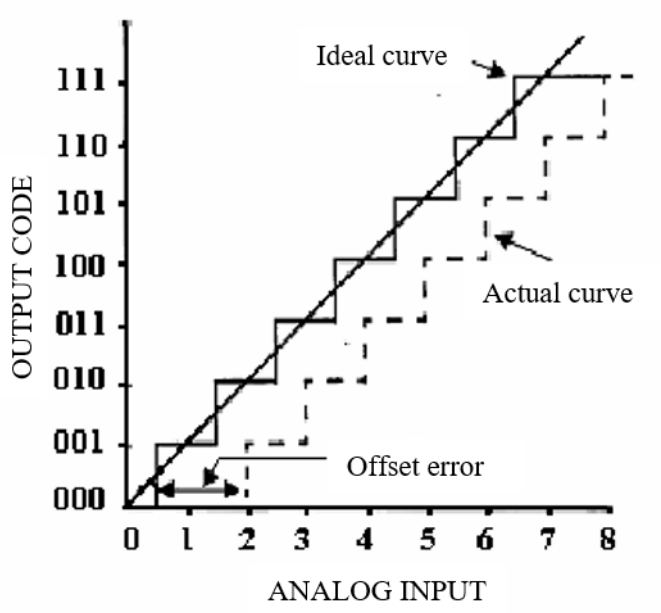



Fig. 2.8 Offset error.

(6) Gain error

The gain error reflects the deviation of the slope of the actual quantization curve of the ADC from the slope of the ideal curve, and it is often expressed in terms of the full-scale gain error, namely shifting the actual quantization curve so that its lowest conversion level is aligned with the lowest conversion level of the ideal curve, and then comparing the deviation of the actual quantization curve with the highest conversion level of the ideal curve. An example of the gain error of the ADC is shown in Fig. 2.9.

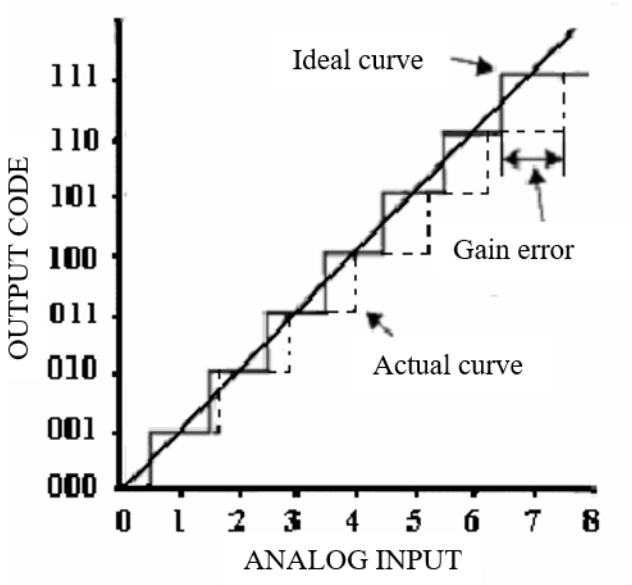



Fig. 2.9 Gain error.

## 2.4 ADC main structure

Nowadays, ADCs have been widely used in various types of System-on-Chips (SoCs), and the requirements of ADCs vary from application system to application system. In order to adopt to the application requirements of different SoCs, various structure types of ADCs have emerged during their continuous development, mainly flash ADCs, pipeline ADCs and SAR ADCs. This section focuses on the operating principles of the above types of ADCs and clarifies their application scope as well as their advantages and disadvantages.

### 2.4.1 Flash ADC

ADC as the core part of analog signal circuit, in many applications, high conversion speed is required, however, flash ADC (fully parallel ADC) has the simplest structure and the fastest conversion rate is often also used.

For an N-bit flash ADC can be represented as Fig. 2.10, which mainly

consists of $2^N$ matching resistors, $(2^N - 1)$ comparators and an encoder. Among then, $2^N$ matching resistors divide the reference voltage $V_{ref}$ into $2^N$ equal-step comparison references, and the comparator compares the input signal with these reference voltages, the voltage difference between each adjacent voltage is $V_{ref}/2^N$ (that is, LSB). The other input of the comparator is connected to the analog input signal. This gives a $2^N$ bit thermometer code by comparing it with each reference voltage, and the final binary code output is completed by a decoder. The flash ADC is a simple structure that requires only one comparison cycle to complete the conversion of the entire ADC, and if the speed of the comparator is guaranteed, the structure can achieve a high conversion rate. However, as the accuracy N increases, the input capacitance increases exponentially, resulting in a smaller input bandwidth. The numbers of comparators and resistors also increase exponentially with N, resulting in this type of ADC requiring a larger chip area, larger power consumption, and even larger input capacitance. Therefore, flash ADCs are often used in low-resolution systems, such as satellite communications, high-speed instrumentation, radar, and video [5-9].

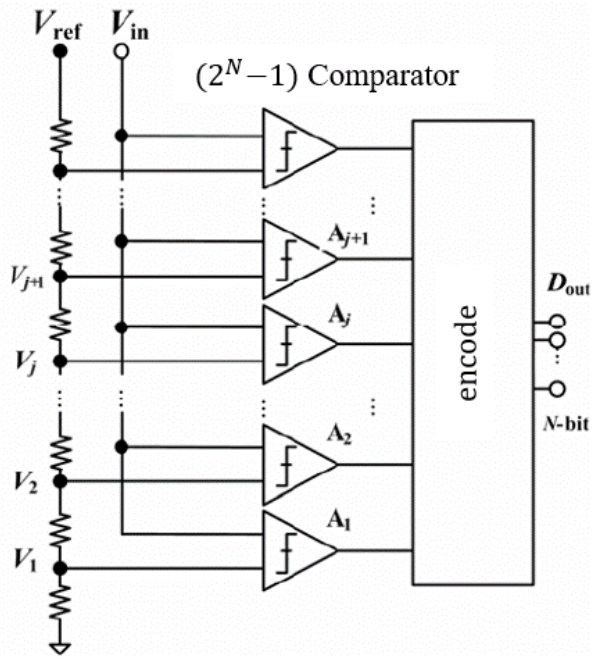



Fig. 2.10 Flash ADC structure.

### 2.4.2 Two-step ADC

The two-step ADC separates the quantization process of higher and lower bits and consists of two ADCs, as shown in Fig. 2.11. It consists of a sample/hold (S/H) circuit, a first-stage ADC, a DAC, a subtractor, and a second-stage ADC. The resolution of the first ADC stage is M-bit, the second stage is L-bit, and the total resolution is (M+L)-bit. The sampled signal is converted to digital by the first stage ADC with high bit (MSBs), and then to analog by the DAC with VB, and the subtractor generates the difference between VA and VB (called the residual). This residual signal is processed by the second ADC stage to obtain a digital output of lower bits (LSBs). However, since S/H, quantization, A/D conversion, D/A conversion, subtraction operations and other operations are performed sequentially, the sampling rate is somewhat limited.

By separating the quantization process of MSBs and LSBs, the two-step structure reduces the number of comparators from $(2^N - 1)$ in the original Flash structure to $(2^M + 2^N - 2)$ and the number of resistors from $2^N$ to $(2^M + 2^N)$. Compared with the flash ADC, the two-step ADC has a reduced operating speed, but has reduced power consumption and chip area, and is often used in video signal acquisition, mobile communication, high-speed portable systems, and so on [10-12].

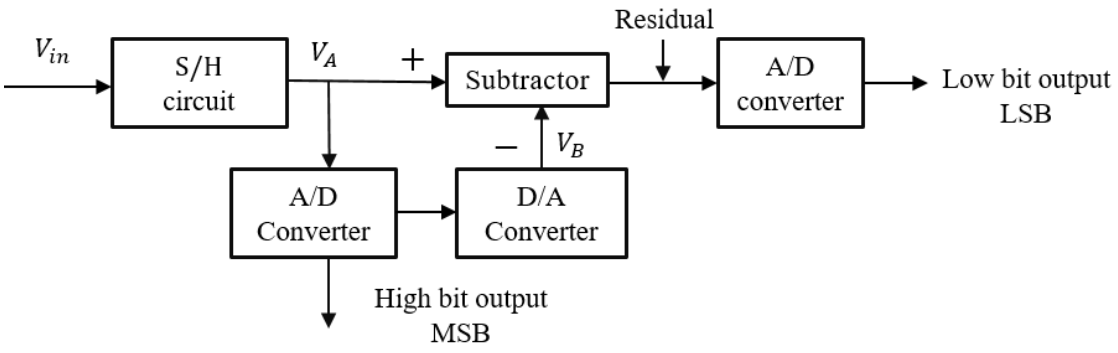



Fig. 2.11 Two-step ADC structure block diagram.

### 2.4.3 Pipeline Converter

The pipeline ADC is based on the two-step ADC principle and consists of several stages cascaded with sub-circuits of similar structure and function. To avoid sampling rate limitation, a sample-hold circuit is added to each stage. Fig. 2.12 shows a block diagram of a K-stage pipeline ADC structure, where each stage 1 to (K-1) contains an S/H, m-bit ADC, m-bit DAC converter, subtractor and amplifier, and the last stage usually uses a flash ADC.

First, the sampled signal from the analog input is converted into an m-bit digital quantity by the first ADC stage, and then this digital quantity is converted into an analog quantity by the DAC, and the residual is obtained

using a subtractor, which is amplified and passed to the next stage for processing.

Since each stage can implement S/H, all stages can work simultaneously. The speed of the pipeline ADC is limited by the conversion speed of each stage and the setup time of the sampling circuit of the next stage. Pipeline ADCs offer conversion rates comparable to flash ADCs with less power and chip area. The main applications are high-speed digital instrumentation, video signal processing, medical imaging, and wireless LAN systems [13,14].

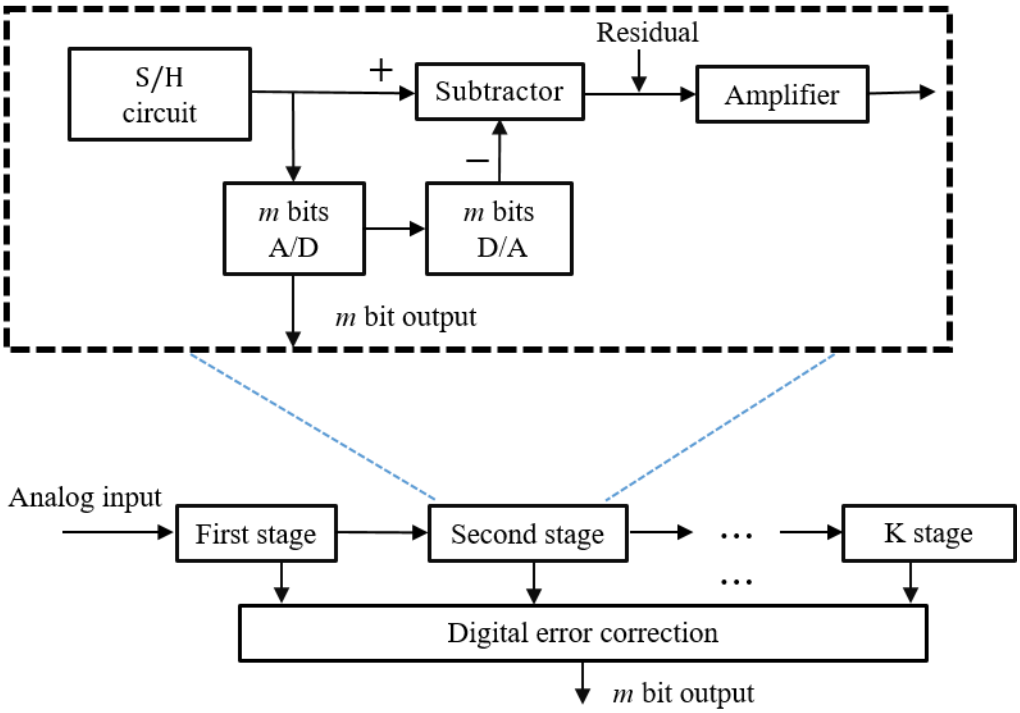



Fig. 2.12 Pipeline ADC structure block diagram.

### 2.4.4 SAR converter

The Successive Approximation Register (SAR) ADC is a common ADC structure with a sampling rate below 5MS/s with medium to high accuracy, and also it belongs to the Nyquist ADC type. Fig. 2.13 shows the

block diagram of the SAR ADC structure [15,16], which mainly includes a S/H circuit, a comparator, a DAC and logic control circuit. The SAR ADC works based on the binary search method by comparing the sampled value $V_{S/H}$ of the input signal with the reference voltage value generated by the D/A conversion network, the logic output from the high bit to the low bit is successively generated.

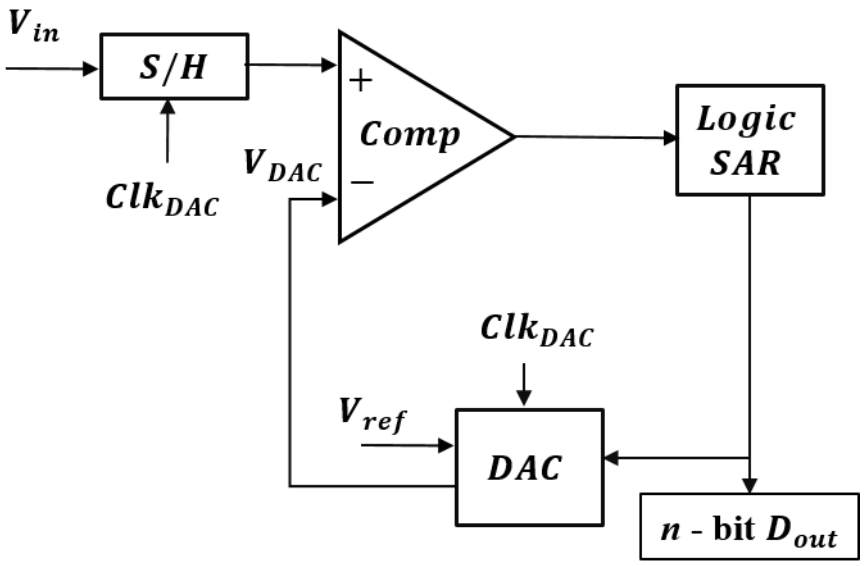



Fig. 2.13 SAR ADC structure block diagram.

Fig. 2.14 shows the workflow diagram of SAR ADC. The SAR ADC first samples and holds the analog signal, compares the magnitude of the input signal $V_{in}$ and the output reference voltage $V_{dac}$ of the capacitor array, and then uses the comparison result to control the switch flip connected to the capacitor array of the DAC to increase or decrease the corresponding voltage value, repeats this operation until the DAC completes the N-bit conversion, and the N-bit digital code stored in the successive approximation register is the final output code, which is complete for the A/D conversion.

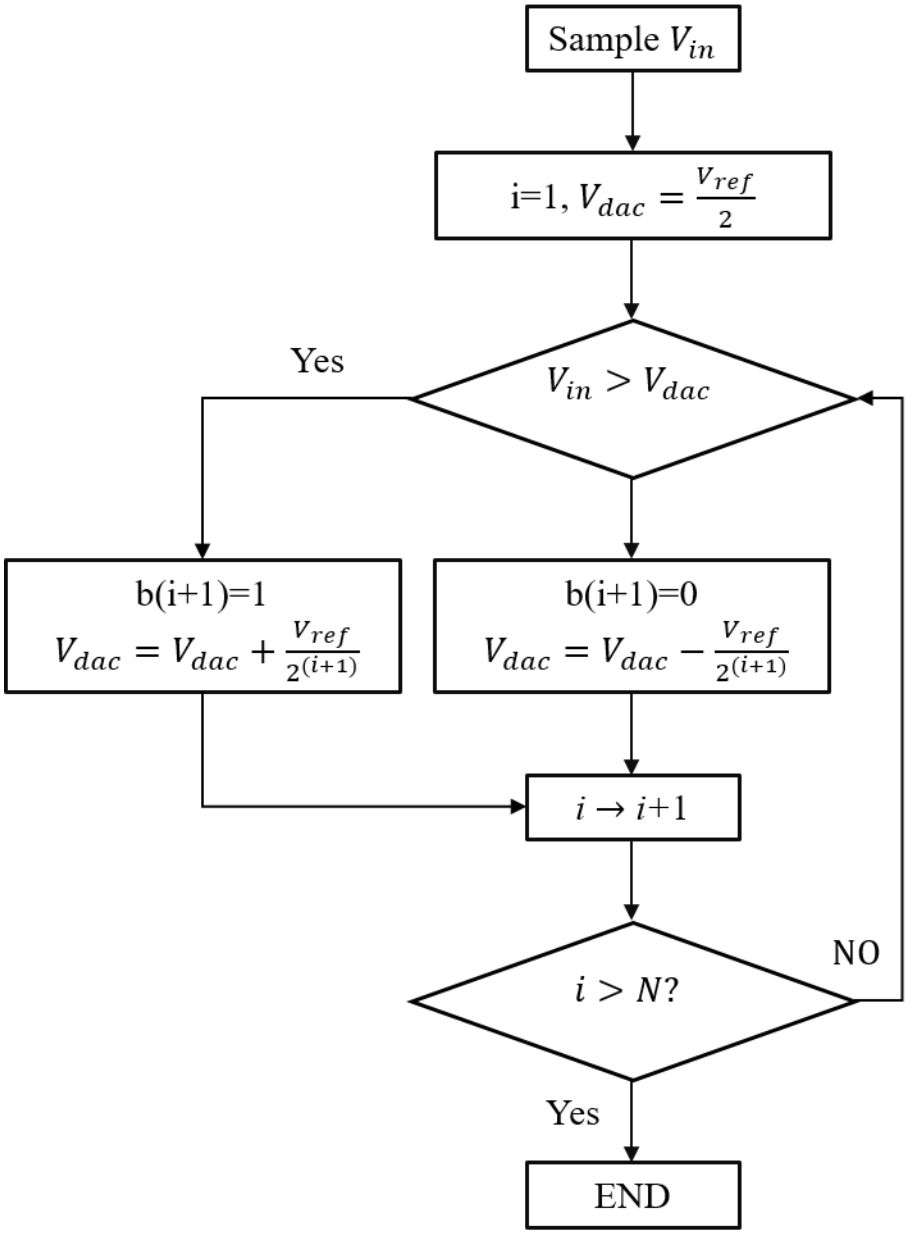



Fig. 2.14 SAR ADC working process diagram.

## 2.5 ΔΣ Converter

ΔΣ ADC is one of the directions of high-speed ADC development in recent years, which can achieve extremely high resolution conversion [17,18]. Traditional ADCs basically follow the Nyquist ADC, according to the Nyquist sampling theorem, where the sampling frequency is higher than

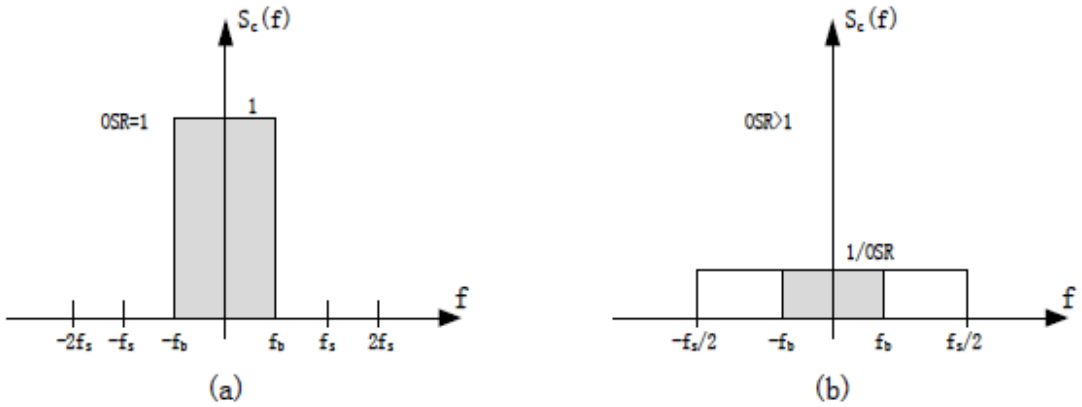
twice the signal bandwidth, and the quantization noise is uniformly distributed in the frequency band of $0 \sim f_s/2$. If the sampling frequency is further increased, then the quantization noise will be uniformly distributed over a larger frequency domain. In other words, if the noise in the high frequency part is filtered out, the signal-to-noise ratio of the ADC can be improved without increasing the accuracy. The noise shaping technique is to move the noise from the low-frequency part to the high-frequency part, and then pass through a low-pass filter to reduce the noise very well. The analog circuit part of the ΔΣ ADC is very simple, but the digital signal processing part is very complex.

### 2.5.1 Oversampling

ADCs applying the oversampling technique show a significant improvement in signal-to-noise ratio compared to Nyquist ADCs. We consider the quantization noise introduced by the oversampling ADC as white noise. After the system has sampled the input signal, part of the noise is distributed outside the signal bandwidth, while the total noise power of the system does not change, which means that the in-band noise power is effectively reduced, which contributes significantly to the improvement of the modulator accuracy. Therefore, the accuracy in the oversampling ADC increases with the sampling rate, but the Nyquist ADC noise is all distributed within the signal band, and the oversampling ADC has its own unique advantage in terms of accuracy. The definition of oversampling rate is as follows:

$$\mathrm{OSR} = \frac{f_s}{2f_b} \tag{2-13}$$

The effect of oversampling rate on the power spectral density of the ADC is shown in Fig. 2.15.




(a) OSR=1 (b) OSR > 1

Fig. 2.15 Influence of OSR on DSP of ADC.

We know that the power spectral density PSD of the quantization noise of the ADC using the oversampling technique, can be expressed as follows:

$$S_e(f) = \frac{\Delta^2/12}{f_s} \tag{2-14}$$

Where, $\Delta^2/12$ is quantized noise power in ADC.

To obtain the quantized noise power, the above equation is integrated for $[-f_b,\ f_b]$:

$$P_e = \int_{-f_b}^{f_b} s_e(f)df = 2f_b.\frac{\Delta^2/12}{f_s} = \frac{1}{OSR}.\frac{\Delta^2}{12} \tag{2-15}$$

The signal-to-noise ratio (SNR) of an oversampling ADC can be written as follows:

$$\text{SNR} = 10log_{10}\left(\frac{3}{2}2^{2N}OSR\right) = 10log_{10}(OSR) + 6.02 + 1.76 \quad (2\text{-}16)$$

The SNR of the Nyquist ADC is:

$$\text{SNR} = 10log_{10}\left(\frac{P_s}{P_e}\right) = 6.02\text{N} + 1.76 \quad (2\text{-}17)$$

We see that the signal-to-noise ratio is proportional to the oversampling rate in a positive logarithmic relationship of 4 to 1, and this proportional relationship creates a bottleneck that relies on increasing the oversampling rate for accuracy improvement.

The relationship among SNR, N and OSR can be clearly seen in Fig. 2.16. The improvement of SNR can be achieved by increasing N and OSR. The higher-order ΔΣ modulator requires much less OSR to achieve the same resolution with the same accuracy.

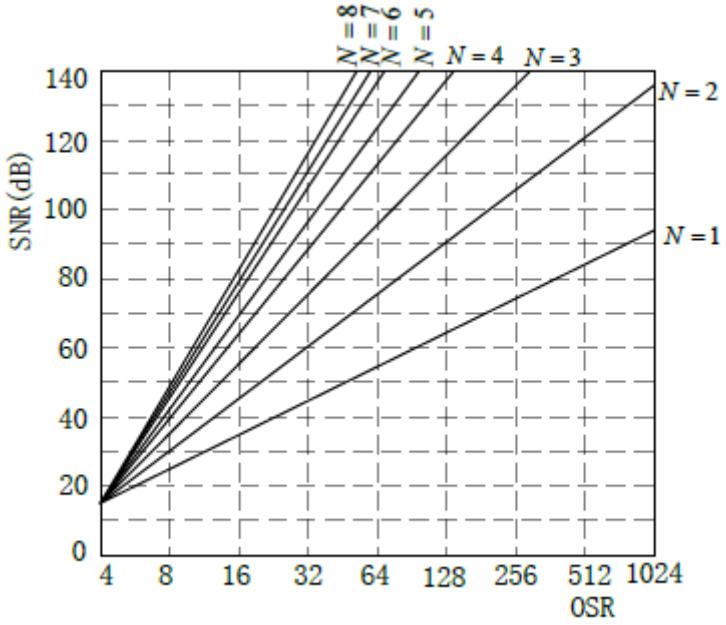



Fig. 2.16 Relationship among SNR, OSR and N.

### 2.5.2 Noise shaping

Noise shaping refers to the shaping of noise through the loop in the process of sampling and quantization of the signal, and does not reduce the introduced noise substantially, but changes the noise distribution on the spectrum by filtering it to separate the in-band quantization noise to the high-frequency band outside the band, from improving the system signal-to-noise ratio and accuracy. The separated out-of-band noise can be filtered out by the digital filter in the successor circuit.

Fig. 2.17 shows a basic modulator structure model with a first-order structure. Where $\mathrm{X(z)}$ is the analog signal to be measured at the system input, $E_q(z)$ is the quantization noise introduced by the system, $\mathrm{H(z)}$ is the transfer function of the integrator and it has a noise shaping function, and $\mathrm{Y(z)}$ is the system output signal.

When quantization noise is not considered, the output signal $\mathrm{Y(z)}$ can be expressed as:

$$\mathrm{Y(z) = H(z)(X(z) - Y(z))} \tag{2-17}$$

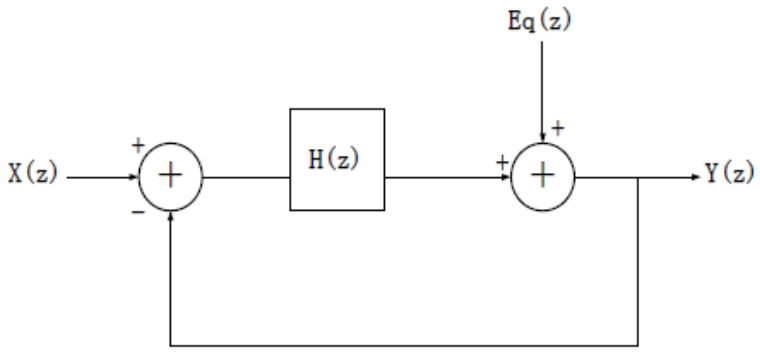



Fig. 2.17 First-order ΔΣ modulator.

The signaling transfer function STF(z) can be expressed as:

$$\mathrm{STF(z)} = \frac{Y(z)}{X(z)} = \frac{H(z)}{1 + H(z)} \tag{2-18}$$

Set the input signal to zero, and the expression of the noise transfer function NTF(z) can be obtained as:

$$\mathrm{NTF(z)} = \frac{Y(z)}{E_q(z)} = \frac{1}{1 + H(z)} \tag{2-19}$$

The transfer function of the entire system can be expressed as:

$$\mathrm{Y(z)} = \mathrm{STF(z)}X(z) + NTF(z)E_q(z) = \frac{H(z)X(z) + E_q(z)}{1 + H(z)} \tag{2-19}$$

When the transfer function H(z) of the integrator has a low-pass characteristic and a large DC gain, the above equation shows that STF(z) is approximately 1 and NTF(z) is approximately infinitesimal, which shows that it has good transmission characteristics in the low frequency range.

### 2.5.3 ΔΣ modulator structures

The basic idea of ΔΣ modulator is oversampling and noise shaping, and the basic structure can be divided into two categories: single loop and cascade (MASH). Different structures of ΔΣ modulators have different characteristics, which are described in the following order:

(1) Single-loop structure

The general structure of a single-loop ΔΣ modulator is shown in Fig. 2.18. The loop filter consists of $L_0$ and $L_1$, which process two inputs, the input signal *u* and the feedback signal *v*. The output y of the loop filter is

used as the input of the quantizer, and *v* is the output of the quantizer. The transfer function from the input signal *u* to *y* is $L_0(z)$, and the transfer function from the feedback signal *v* to *y* is $L_1(z)$, and the transmission process of the system is represented by the z transformation as shown in following equation:

$$\begin{cases} Y(z) = L_0(z)U(z) - L_1(z)V(z) \\ V(z) = Y(z) + E(z) \end{cases} \tag{2-20}$$

V(z) can generally be expressed as follows:

$$\mathrm{V(z)} = \mathrm{STF(z)U(z)} - \mathrm{NTF(z)E(z)} \tag{2-21}$$

Then, STF(z) and NTF(z) can be expressed as follows:

$$\mathrm{STF(z)} = \frac{L_0(z)}{1 + L_1(z)} \tag{2-22}$$

$$\mathrm{NTF(z)} = \frac{1}{1 + L_1(z)} \tag{2-23}$$

The *z* domain of the integrator can be expressed as:

$$\mathrm{H(z)} = \frac{z^{-1}}{1 - z^{-1}} \tag{2-24}$$

We take the example of a first-order single loop ΔΣ modulator. Let, $L_0(z) = L_1(z) = H(z)$, the following equation can be obtained:

$$\mathrm{STF(z)} = z^{-1} \tag{2-25}$$

$$\mathrm{NTF(z)} = 1 - z^{-1} \tag{2-26}$$

Then the output responses of u and v are superimposed to obtain the

input of the quantizer as:

$$\mathrm{Y(z)} = z^{-1}X(z) + (1 - z^{-1})E(z) \qquad (2\text{-}27)$$

This type of modulator is called a first-order modulator, and the order refers to the order of noise shaping, as shown in Fig. 2.19.

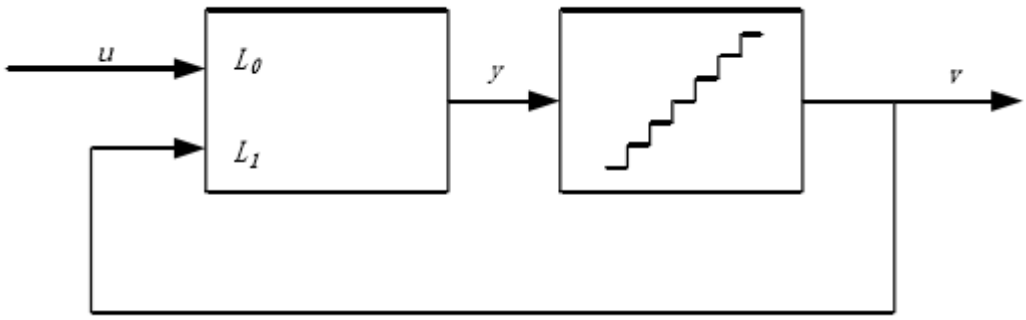
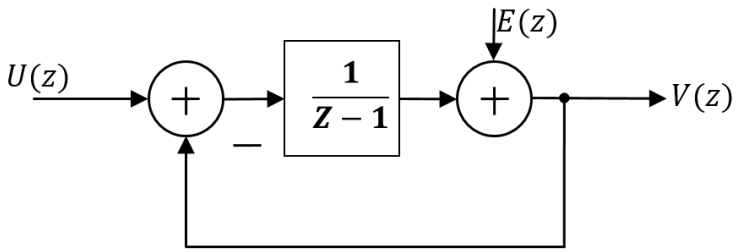
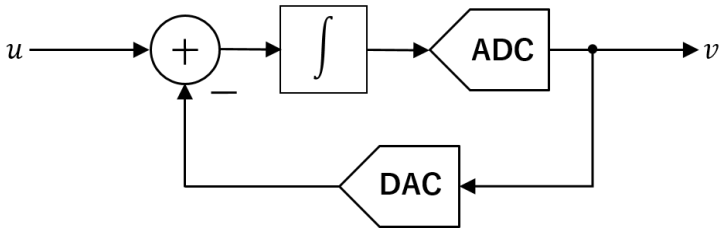



Fig. 2.18 General structure of single-loop ΔΣ modulator.




(a) Linearized z-domain model.




(b) First-order ΔΣ modulator structure.

Fig. 2.19 First-order ΔΣ modulator.

(2) Cascade (Mash) structure

For higher-order single-loop ΔΣ modulators, the system stability problems are easily encountered due to the influence of factors such as non-linearity, and the performance of actually designed modulator can be significantly degraded compared to the ideal modulator. In order to solve the problems that arise in higher-order ΔΣ modulators, one approach is to use a cascaded ΔΣ modulator structure [19, 20], as shown in Fig. 2.20. A cascaded ΔΣ modulator is made up of a number of low order single loop ΔΣ modulators cascaded together. As can be seen from the diagram, all the feedback loops are only at each level of the ΔΣ structure, there is no feedback from stage to stage, and only noise is transmitted from stage to stage. So if the ΔΣ modulator of each low order single loop is stable, then the ΔΣ modulator of the whole cascade structure is also stable. That is, the stability of the higher stages of the cascade is directly based on the stability of the lower single-loop stages, and the high-order ΔΣ modulator is easier to design and implement.

For the entire output of the cascade modulator, the output of each modulator stage is logically summed by the transfer function $H_1(z)$ and $H_2(z)$ for noise cancellation, so that only the noise of the last stage is transferred to the output, but there is a certain mismatch between the analog part of the modulator circuit and the digital circuit of $H_1(z)$ and $H_2(z)$, which affects the performance. Therefore, the cascade structure has high requirements for the matching of the device and is not as easy to implement without a single-loop structure.

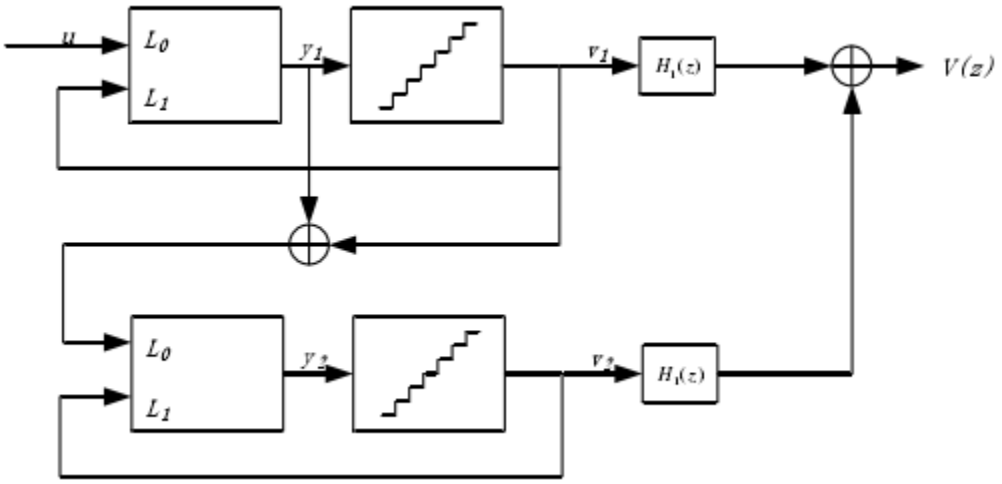



Fig. 2.20 MASH ΔΣ structure.

## 2.6 Summary

This chapter first introduces the static and dynamic characteristics of ADCs, and briefly explains the definitions of these characteristics, then introduces each of the five common types of ADCs, briefly describes their respective structures, including the circuit modules and the functions of each module, and briefly describes their operating principles. It also analyzes the advantages of oversampling and noise shaping of the ΔΣ ADC itself. Table 2.1 summarizes the performance of the previously described ADCs.

Table 2.1 Typical Architecture ADC Performance Summary

| ADC structure type | Sampling rate | Resolution (accuracy) | Power | Application |
|---|---|---|---|---|
| Flash | GS/s | 6-8 bit | High | High-speed system, etc. |
| Two-step | MS/s | 8-12 bit | Medium | Wireless communication, medical imaging |
| Pipeline | MS/s | 10-14 bit | Lower | High-speed video equipment |
| SAR | KS/s | 8-16 bit | Low | Industrial equipment, data survey |
| ΔΣ | KS/s | 16-24 bit | Medium | Audio, precision measurement |

# References


[1] T. L. Lago, "Digital Sampling According to Nyquist and Shannon", Sound and Vibration vol. 36, no. 2, pp. 20-22 (Feb. 2002).

[2] P. Cruz, N. B. Carvalho and K. A. Remley, "Evaluation of Nonlinear Distortion in ADCs Using Multi-Sines", IEEE MTT-S International Microwave Symposium, Atlanta, GA (Jun. 2008).

[3] J. M. D. Pereira, A. C. Serra and P. S. Girao, "Dithered ADC Systems in the Presence of Hysteresis Errors", 16th IEEE Instrumentation and Measurement Technology Conference, Venice, Italy (May. 1999).

[4] M. Abuelma'atti, "Effect of Nonmonotonicity on the Intermodulation Performance of ADCs", IEEE Transactions on Communications vol. 33, no. 8, pp. 839-843 (Aug. 1985).

[5] B. Razavi, M. B. Tavakoli, and F. Setoudeh, "Approach for Low Power High Speed 4-Bit Flash Analogue to Digital Converter", IET Circuits, Devices & Systems, vol. 14, no. 4, pp. 425-431 (Jul. 2020).

[6] H. Fan, J. Yang, Y. Cen and Q. Feng, "Optimization of High Precision SAR ADC Used in the Remote Sensing Technology", IEEE International Geoscience and Remote Sensing Symposium, Waikoloa, HI (Sept. 2020).

[7] S. Kumar and C. Nagesh, "Design of a Two-Step Low-Power and High-Speed CMOS Flash ADC Architecture," 24th International Symposium on VLSI Design and Test, Bhubaneswar, India (Jul. 2020).

[8] S. Firojuddin, S. Pal, and P. Pain, "Design and Comparative Analysis of Low-Power, High-Speed, 3-bit Flash ADC for Biomedical Signal Processing Using 45-nm CMOS Technology", Computational Advancement in Communication Circuits and Systems. Springer, Singapore, vol. 575, pp. 331-342 (Jan. 2020).

[9] Joshi, Atul, Sachin Kashyap, and Vajayeendra Rao, "Application Specific Integrated Circuit (ASIC) With Low Power Digitizer (ADC) for Space Imaging Applications", Sensors and Systems for Space Applications XIV. International Society for Optics and Photonics, Vol. 11755 (Apr. 2021).

[10] J. Doernberg, P. R. Gray, and D. A. Hodges, "A 10-bit 5-Msample/s CMOS Two-Step Flash ADC." IEEE Journal of Solid-State Circuits vol. 24, no. 2, pp. 241-249 (Apr. 1989).

[11] K. L. Krishna, Y. M. A. Al-Naamani, and K. Anuradha, "A High Speed Two Step Flash ADC", International Conference on Soft Computing Systems, Springer, Singapore (Sept. 2018).

[12] A. Cremonesi, F. Maloberti, G. Torelli and C. Vacchi, "An 8-Bit Two-Step Flash ADC for Video Applications", IEEE Custom Integrated Circuits Conference, San Diego, CA, USA (May. 1989).

[13] J. Arias, V. Boccuzzi, L. Quintanilla, L. Enriquez, D. Bisbal, M. Banu, J. Barbolla, "Low-Power Pipeline ADC For Wireless LANs", IEEE Journal of Solid-State Circuits, vol. 39, no. 8, pp. 1338-1340, (Aug. 2004).

[14] M. Trojer, M. Cleris, U. Gaier, T. Hebein et. al., "A 1.2V 56mw 10 Bit 165Ms/S Pipeline-ADC for HD-Video Applications", 34th European Solid-State Circuits Conference, Edinburgh, UK (Sept. 2008).

[15] C. Liu, C. Kuo and Y. Lin, "A 10 Bit 320 MS/S Low-Cost SAR ADC for IEEE 802.11ac Applications In 20 Nm CMOS", IEEE Journal of Solid-State Circuits, vol. 50, no. 11, pp. 2645-2654 (Nov. 2015).

[16] P. Harikumar, J. J. Wikner and A. Alvandpour, "A 0.4-V Subnanowatt 8-nit 1-kS/s SAR ADC in 65-nm CMOS for Wireless Sensor Applications", IEEE Transactions on Circuits and Systems II: Express Briefs, vol. 63, no. 8, pp. 743-747 (Aug. 2016).

[17] S. Pavan, N. Krishnapura, R. Pandarinathan and P. Sankar, "A Power Optimized Continuous-Time ΔΣ ADC for Audio Applications", IEEE Journal of Solid-State Circuits, vol. 43, no. 2, pp. 351-360 (Feb. 2008).

[18] O. Shoaei, "Continuous-Time ΔΣ ADCs for High Speed Applications", Ph. D. Dissertation, Carleton University, Ottawa, Canada (1996).

[19] N. Maghari and U. Moon, "Multi-Loop Efficient Sturdy MASH ΔΣ Modulators", IEEE International Symposium on Circuits and Systems, Seattle, WA, USA (May. 2008).

[20] J. Silva, U. Moon and G. C. Temes, "Low-Distortion ΔΣ Topologies for MASH Architectures", IEEE International Symposium on Circuits and Systems, Vancouver, BC, Canada (May. 2004).

# Chapter 3

# LIMIT CYCLE SUPPRESSION TECHNIQUE USING RANDOM SIGNAL IN ΔΣ D/A MODULATOR

## 3.1 Abstract

The ΔΣ digital-to-analog converter (ΔΣDAC) generates a DC/low frequency analog signal with high resolution and high linearity, whereas there is a limit cycle problem where spurious components are periodically generated in the output signal for the small amplitude input, due to the nonlinear operation of the modulator. This paper studies a limit cycle suppression technique in the ΔΣ DA modulator. It is shown in simulations that the investigated method can reduce the limit cycle, while keeping the overall linearity, thanks to the fact that the random noise is noise-shaped. We show that these are valid for various types of the modulators: low-pass (LP), high-pass (HP), band-pass (BP) and multi-BP types.

## 3.2 Introduction

A DAC is widely used to generate an analog signal from a digital signal by sampling and quantization (Fig. 3.1). The ΔΣ DAC is commonly used in audio applications, cellular phone technology and high-end stereo systems, because it has properties of low cost, limited bandwidth, low power and high resolution/high linearity. However, the ΔΣ DA modulator suffers from a limit cycle problem when its input amplitude is very small [1].

In order to overcome the problem mentioned above, we investigate a limit cycle suppression method, which adds a digital random noise to one of the comparator inputs for various types of modulators [5]: the loop filter H(z) in Fig. 3.1 is defined according to the modulator type as follows:

(1) For low-pass (LP) type:

$$H(Z) = \frac{-z^{-1}}{1+z^{-1}} \tag{3-1}$$

(2) For high-pass (HP) type

$$H(Z) = \frac{z^{-1}}{1-z^{-1}} \tag{3-2}$$

(3) For band-pass (BP) type

$$H(Z) = \frac{-z^{-2}}{1+z^{-2}} \tag{3-3}$$

(4) For multi-BP type I

$$H(Z) = \frac{-z^{-N}}{1+z^{-N}} \tag{3-4}$$

(5) For multi-BP type II

$$H(Z) = \frac{z^{-N}}{1-z^{-N}} \tag{3-5}$$

The SNDR is not degraded even though the random noise is added, because it is added at the end of the feedforward path in the modulator and hence the added noise is so-called noise shaped.

Our MATLAB simulation results confirm that the limit cycles are suppressed in 10,14,16,18-bit cases, for low-pass (LP), band-pass (BP) and

multi-BP type modulators, while their good linearity is kept.

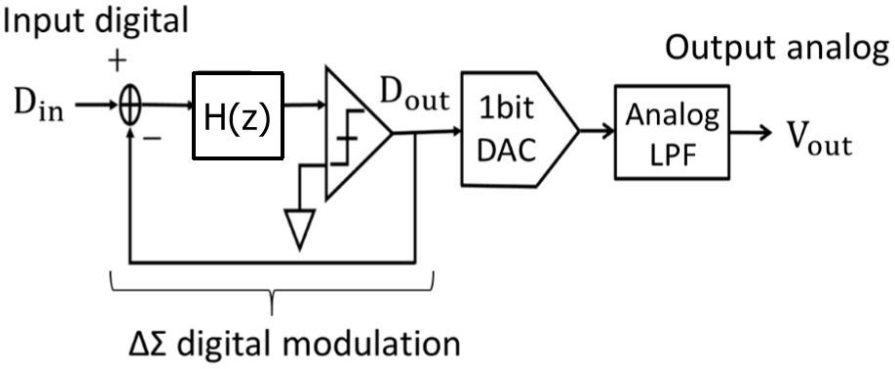



Fig. 3.1 Block diagram of the first-order ΔΣ DA converter.

## 3.3 ΔΣ Modulator

### 3.3.1 ΔΣ Modulator configuration and operation

The ΔΣ DA modulator consists of all digital circuits with feedback configuration with the digital input ($D_{IN}$) and the digital output ($D_{OUT}$), using a loop filter H(z) (e. g., an integrator for LP type) and a comparator (Fig. 3.1). The error signal is accumulated at the integrator, and its output is compared with zero by the comparator. The integrator then adds the output of this summing node to a value stored from the previous integration step. The comparator outputs a logic 1 if the integrator output is greater than or equal to zero, and a logic 0 otherwise. The 1-bit DAC feeds the output of the comparator back to the summing node: + MSB for logic 1 and - MSB for logic 0. This feedback tries to keep the integrator output at zero by making the ones and zeros output of the comparator to be equal to the digital input. It is known in [4] that the output power spectrum is noise-shaped; the quantization noise is reduced at low frequency region while increased at high frequency.

### 3.3.2 Quantization noise

The quantization error is a difference between the digital integrator output and its quantized output. For most input signals, power of the quantization noise is calculated as δ2/12 (δ is quantization step) in the frequency range between zero to Nyquist frequency fs/2 (fs: sampling frequency) [4].

### 3.3.3 Random signal (Dither signal)

For the original dither signal added to the delta sigma DA conversion is shown in Fig. 3.2. However, this method sacrifices the input range; components of the dither signal enter the signal band, thus reducing the SNR of ΔΣ DA modulator.

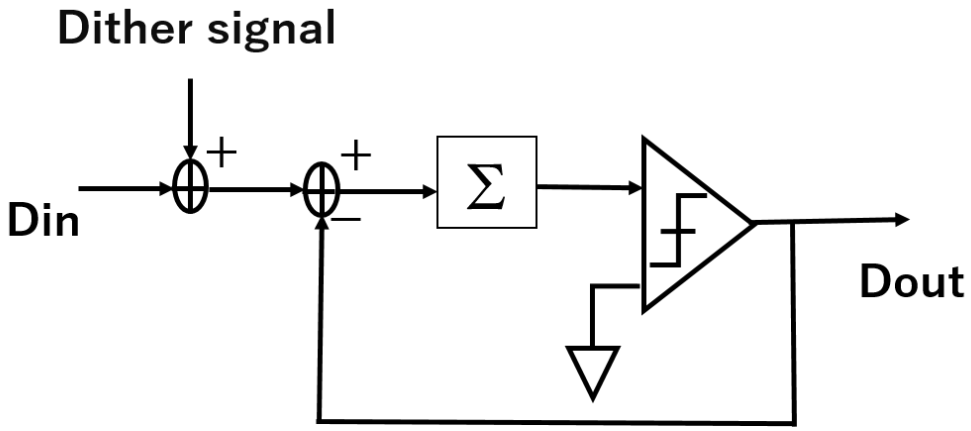



Fig. 3.2 Original dithering method, where the dither signal is added to ΔΣ DA modulator input

Our proposed technique applies a digital random signal to one of the comparator inputs (Fig. 3.3). The proposed method can change the demerits of the original method. Commonly the dither signal can be generated by oscillator, or PRNG (pseudo random number generator) such as LCG (linear

congruential generator) or Mersenne Twister.

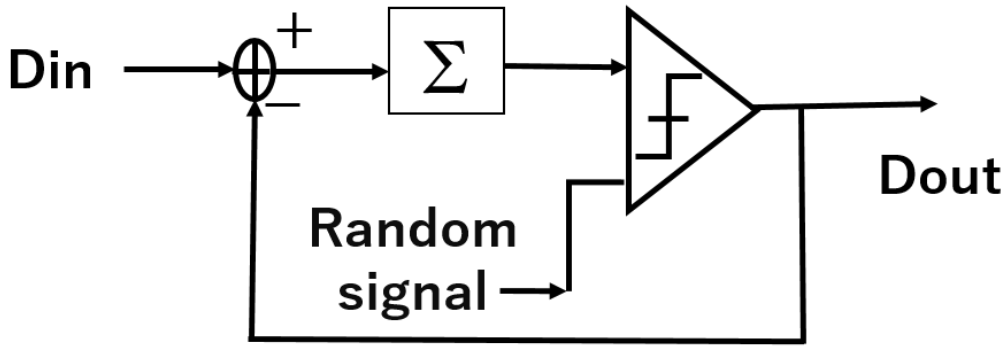



Fig. 3.3 Investigated ΔΣ DA modulator with random signal.

Dithering, or adding random offsets to the comparator input, represents a special case of limit cycle disturbance, since it does not directly influence the integrator output. The only way in which dither can break up a limit cycle is by changing the sign of the input to the comparator, causing it to create a bit-flip in the limit cycle output. As its result, a limit cycle is broken up [3].

## 3.4 MATLAB simulation results

### 3.4.1 Limit cycle suppression (10 Bit case)

We consider the case that the range of the random signal is within -2 ~ +2 (its discussion is included in Section. 3.2). We have checked the limit cycle reduction with simulation. In addition, we have also checked the number of 1's at the modulator output, for DC signal generation is the same in both cases with and without a random signal The linearity is confirmed with this simulation result (Fig. 3.4).

Fig. 3.6 shows simulation results for the DC input (Din) of 0.1. We see that the limit cycle with random signal (Fig. 3.5(b)) is lower than that without random signal (Fig. 3.5(a)), and also that Spurious Free Dynamic Range

(SFDR) [6] with random signal (22.1dB) is much higher than that without random signal (5.39dB). We have used the same simulation method to compare SFDRs with and without random signal and we see in Fig. 3.6 that it is improved for all range of the DC input (Din).

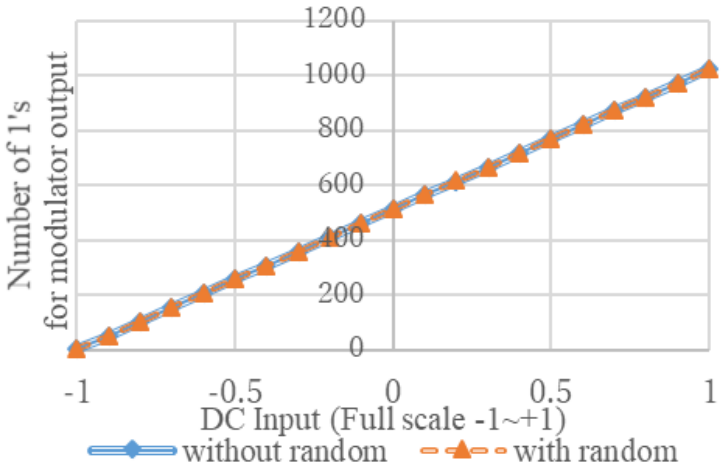
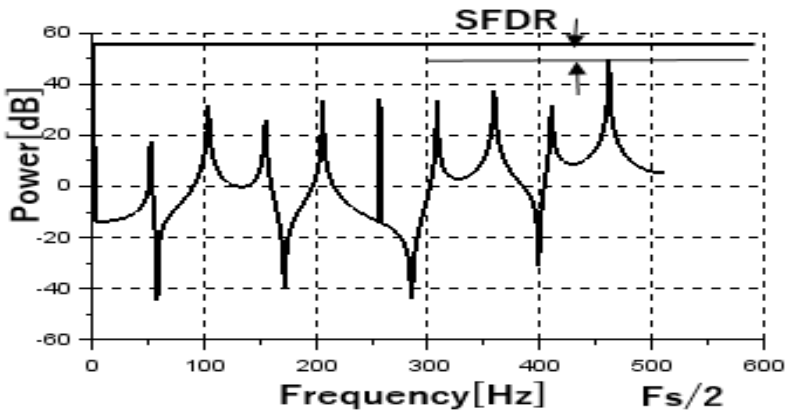
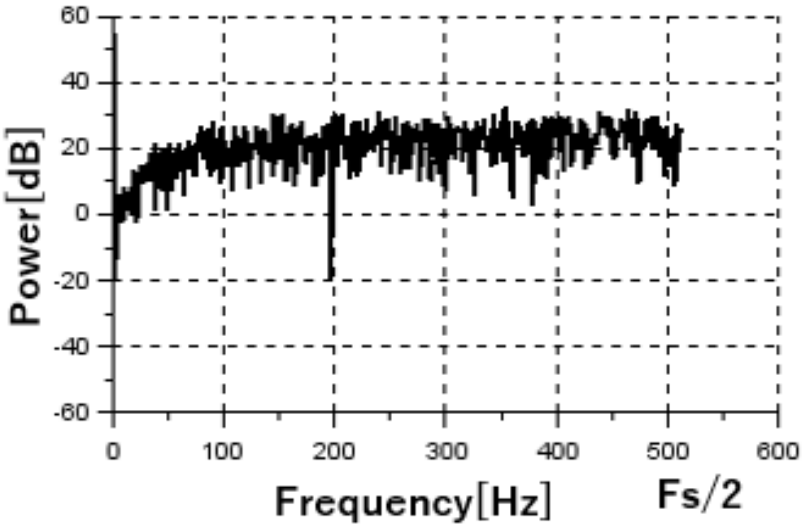



Fig. 3.4 Number of 1's for modulator output with $2^{10}$data.

(The two graphs match exactly because number of 1's is the same.)




(a) Without random signal




(b) With random signal

Fig. 3.5 Power spectrum of ΔΣ DA modulator output in the case that the DC input (Din) is 0.1

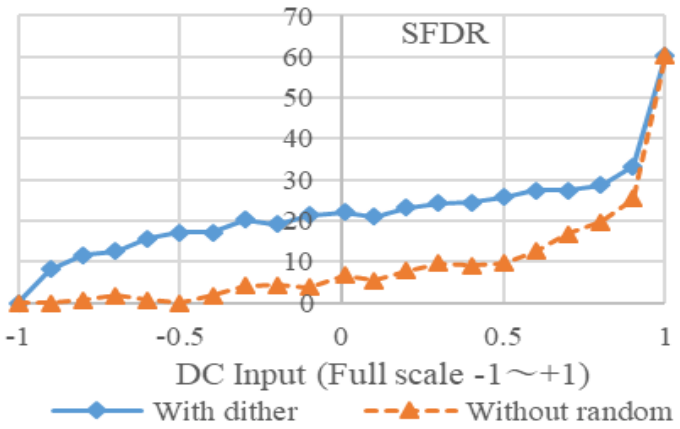



Fig. 3.6 SFDR comparison with and without random signal

### 3.4.2 Random signal range with finite word length

Here we consider the first-order ΔΣ DA modulator. Random signals between -1~+1, -2~+2, and -3~+3 are considered. The comparison results of the SFDR simulation are shown in Fig. 3.7. The ideal result can be obtained

at -2~+2 of random signal.

When the random signal is within -2~+2, we consider its word length. Suppose that a fractional number 1.7824531023... is given. In Fig. 3.8, “precision is 3” means that 1.782 is considered, “precision is 4” means 1.7824 and “precision is 5” means 1.78245. Our simulation results show that changing the precision or the word length does not affect the SFDR at all (Fig. 3.8).

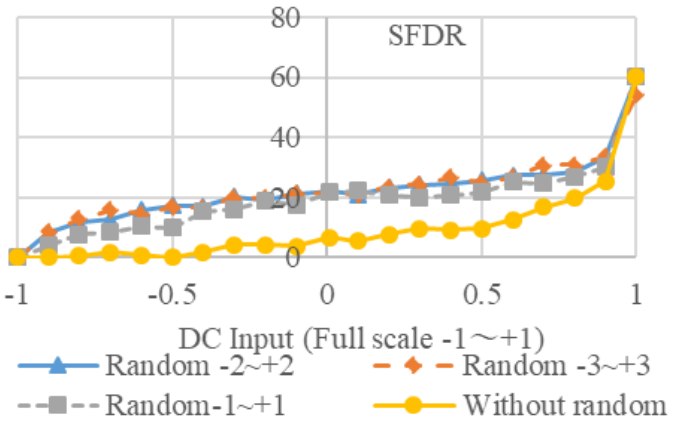



Fig. 3.7 Random signal range and SFDR

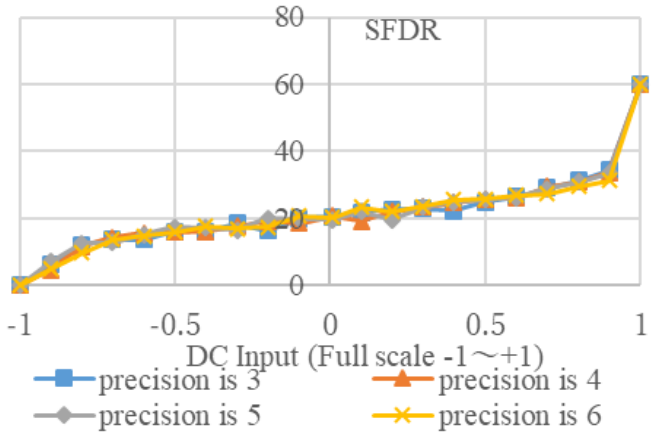
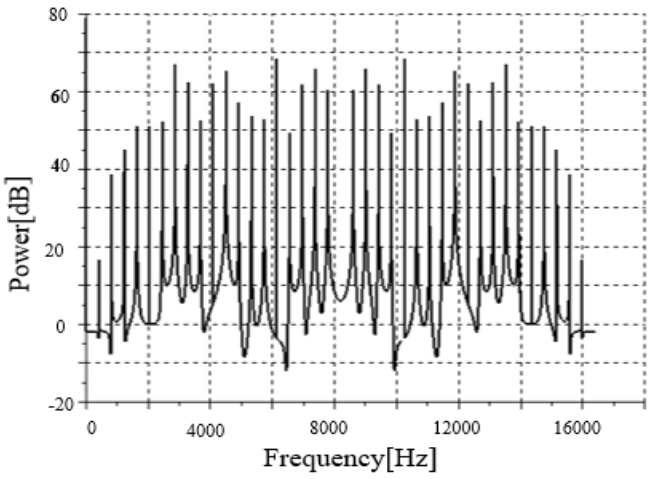



Fig. 3.8 Random signal precision and SFDR.

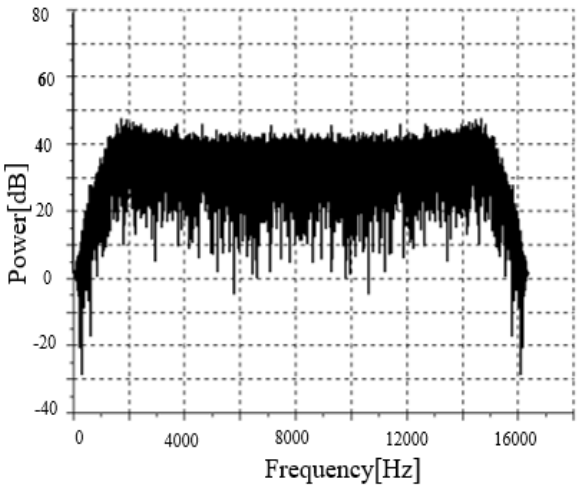
### 3.4.3 Study on reduced circuit of the limit cycle for BP modulator (14, 16, 18-bit cases)

Sections 3.1 and 3.2 describe the validation of the investigated algorithm for the first-order LP ΔΣ modulator. Here we consider the case of the second-order BP ΔΣ modulator.

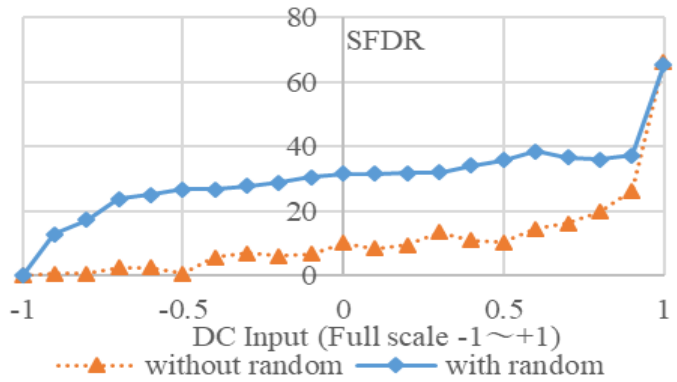
We have simulated the in 14, 16, 18-bit cases for 2-band, 4-band band-pass (BP) modulators, and their results are shown in Figs. 3.9, 3.10 and 3.11. We see in Figs. 3.9 (d), 3.10 and 3.11 that the SFDR is improved for the DC input of full scale between -1 and +1. Fig. 3.9 (d) shows the difference of the modulator output 1's numbers with and without the random signal in 14-bit case. We see that the proposed circuit maintains the good DC linearity because the difference number of 1's is within a ±1 range. We also see in Figs. 3.10 and 3.11 that the linearity is maintained in 16-bit and 18-bit cases. Fig. 3.12 show a second-order multi-BP ΔΣ modulator case (14-bit case). Fig. 3.13 shows the results of the SFDR simulation with different resolutions for the same dithered signal. The comparison of the results shows that as the resolution increases, the linearity follows.



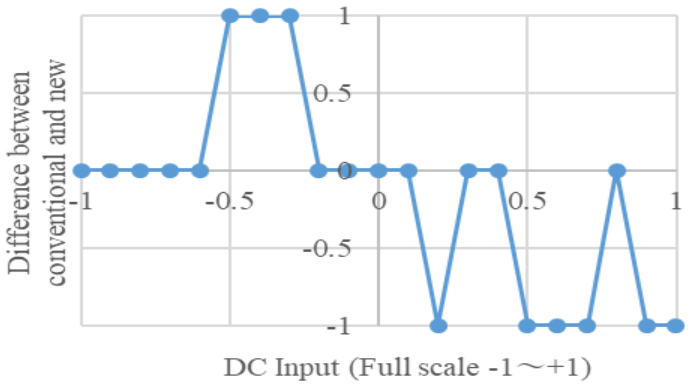

(a) Without random signal




(b) With random signal.




(c) SFDR comparison.




(d) Linearity

Fig. 3.9 Simulation results in 14-bit case.

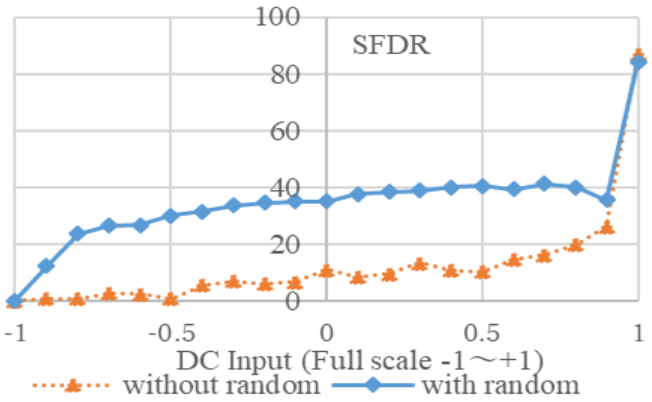



Fig. 3.10 SFDR simulation results in 16-bit case.

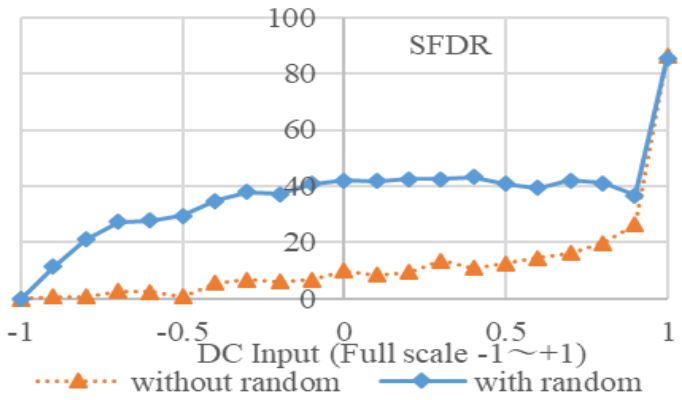
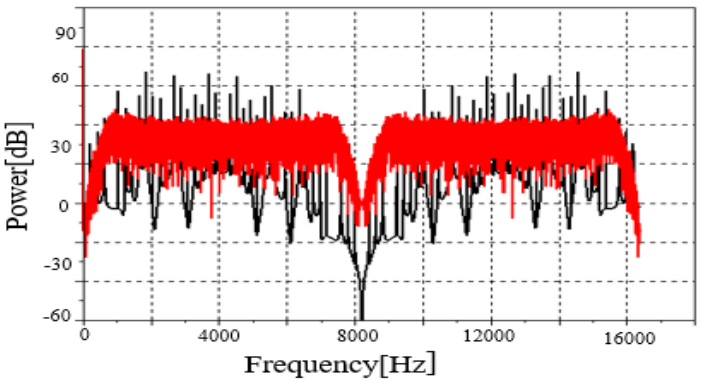
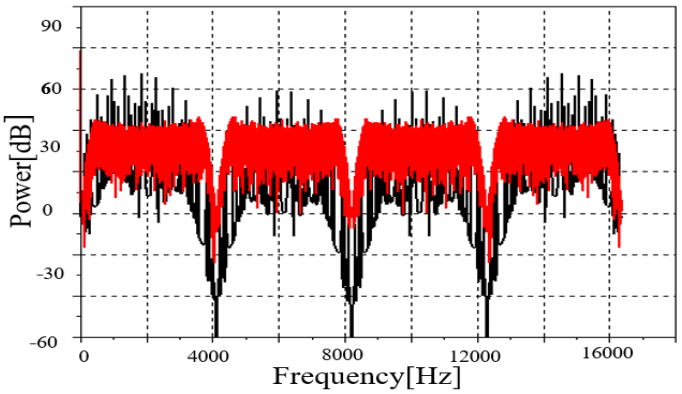



Fig. 3.11 SFDR simulation results in 18-bit case.




(a) 2-band BP modulator.




(b) 4-band BP modulator.

Fig. 3.12 Modulator output power spectrum comparison of without (black) and with random signal (red) for second-order complex BP ΔΣ modulators.

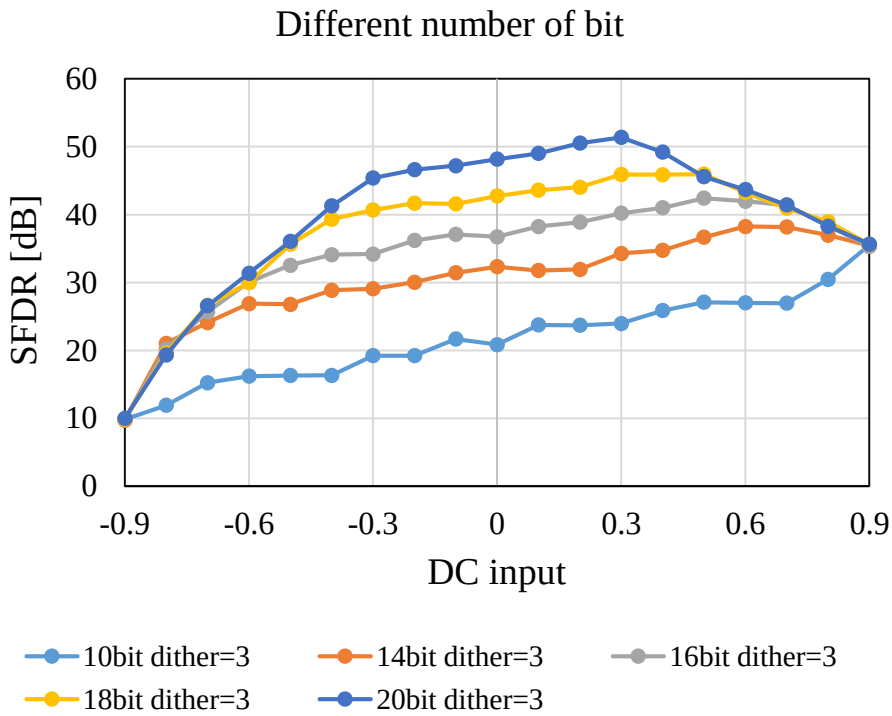



Fig. 3.13 SFDR with different resolutions for the same dithered signal.

## 3.5 Summary

We have investigated a limit cycle suppression technique using a

random signal in the ΔΣ DA modulator. It uses the random signal to added to one input of the comparator in the ΔΣ DA modulator. We have used extensive MATLAB simulation and observed the following:

(a) Limit cycles are reduced and the SFDR is improved.

(b) The overall linearity of the ΔΣ DA modulator is maintained; this is because the added random noise is at the end of the feedforward path and it is noise-shaped.

(c) The above statements are valid for all LP, HP, BP and multi-BP type modulators.

(d) Simple circuit implementation equivalent to Fig. 1 is next challenge. Also this technique can be applied for the ΔΣ AD modulator.

# References


[1] J. Kojima, Y. Arai, H. Kobayashi, “Limit Cycle Suppression Technique Using Digital Dither in ΔΣ DA Modulator”, 13th IEEE International Conference on Solid-State and Integrated Circuit Technology, Hangzhou, China (Oct. 2016)

[2] D. Reefman, J. Reiss, et. al., “Description of Limit Cycles in Feedback ΔΣ Modulators”, IEEE Transactions on Circuits and Systems I: Regular Papers, Vol. 52, no. 6, pp. 1211-1223 (Jul. 2005).

[3] Y. Koyano, et. al., “Recording and Playback System of High Speed Single-bit Direct Quantized Signal with Dither”, Western Pacific Acoustics Conference, Tokyo, Japan (Dec. 2015).

[4] R. Schreier, G. C. Temes, Understanding ΔΣ Data Converters, IEEE Press (2005).

[5] M. Murakami, H. Kobayashi, S. N. Bin Mohyar, O. Kobayashi, T. Miki and J. Kojima, "I-Q Signal Generation Techniques for Communication IC Testing and ATE Systems," IEEE International Test Conference, Fort Worth, TX (Nov. 2016).

[6] R. Plassche, CMOS Integrated Analog-To-Digital and Digital-To-Analog Converters, 2nd Edition, Kluwer Academic Publisher (2003).

# Chapter 4

# SHORT-TIME INL TESTING METHODOLOGY FOR HIGH-RESOLUTION ΔΣ ADC

## 4.1 Abstract

This chapter describes a mass production testing methodology for integral nonlinearity (INL) of a high precision ΔΣ analog-to-digital converter (ADC) in short time. We consider its INL testing by separating its analog and digital parts: ΔΣ AD modulator and digital filter. The digital filter can be tested with the scan-path method. For the AD modulator part, its nonlinear curve of the DC input-output characteristics can be obtained using a DC input varying with a fine step, but it takes an enormously long time; it is not practical for mass production testing. So we consider a polynomial model of the ΔΣ AD modulator input-output characteristics and estimate its coefficient values from the fundamental and harmonics power by applying a cosine input and obtaining the modulator 1-bit output power spectrum with FFT. Its INL can be estimated from the coefficients accurately when the modulator I/O characteristics is continuous. Our simulation and experimental results show that significant testing time reduction can be achieved with the proposed method.

## 4.2 Introduction

In recent years, Internet of Things (IoTs) has attracted much attention, and the testing of IoT-related devices in short time with high quality has

become more important at mass production shipping for the IoT system reliability [1-3]. This chapter focuses on high-resolution low-sampling-rate ΔΣ ADCs, which are widely used with sensor interface circuits, such as air flow, temperature, pressure and strain gauge sensors as well as communication circuits [4-9]. However, its integral nonlinearity (INL) testing takes extraordinary long time; for example, let us consider the case of a 7 sample-per-second (sps) 24-bit ΔΣ ADC and 4 samples for each code used in its INL testing. Then its testing takes 111 days, which is not acceptable at all because the reasonable testing time is 1 second for 1 US dollar chip.

Therefore, in most cases, the INL testing for the ΔΣ ADC is omitted at mass production shipping. However, recently high quality and high reliable systems are demanded. Then we have developed its INL testing algorithm with drastically reduced testing time as well as keeping good testing accuracy, and here we present its algorithm as well as simulation and experimental verifications.

## 4.3 ΔΣ ADC

The ΔΣ ADC is composed of a ΔΣ AD modulator in the analog section and a digital filter (decimator) in the subsequent stage [4-9]. The ΔΣ AD modulator performs ΔΣ modulation for the analog input with oversampling, so that noise shaping for the quantization noise is realized. Then 1-bit digital data stream is provided as the modulator output, which is fed to the following digital filter for low-pass filtering and decimation; its output is the entire ADC digital output (Fig. 4.1).

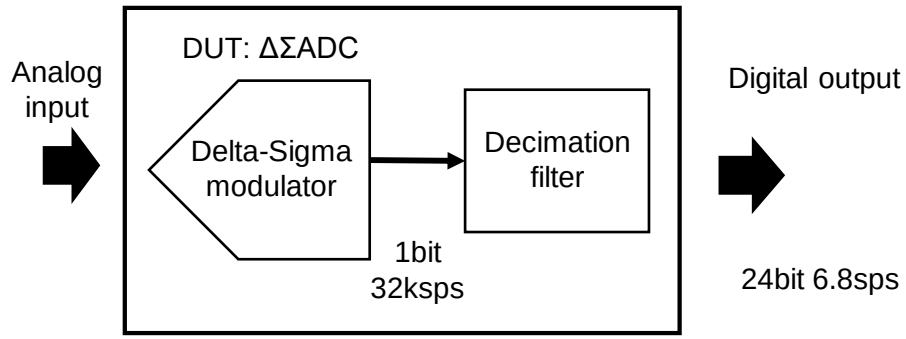



Fig. 4.1 Configuration of a ΔΣ ADC.

## 4.4 Proposed ΔΣ ADC linearity test method

We consider a 7-sps 24-bit discrete-time ΔΣADC for the target application. Notice that only 7 digital output data can be obtained in 1 second, and hence the direct INL testing is not acceptable at all (Fig. 4.2). Hence, we consider here to observe the 32-ksps 1-bit data stream of the ΔΣAD modulator for the INL testing. Notice also that the INL of the overall ΔΣADC is determined only by the ΔΣAD modulator, and the digital filter does not affect the overall ADC INL if it is well-designed and functional (i.e., without any catastrophic faults). Then we propose the following INL testing method (Fig. 4.3):

(1) Separate the AD modulator and the digital filter parts, and test them individually.

(2) The digital filter part is tested by the scan path method whether there are fatal faults. Notice that the digital filter part does not cause the overall ΔΣAD linearity deterioration unless it is faulty.

(3) The 1-bit output data stream of the ΔΣAD modulator is externally outputted through a test pin in test mode (Fig. 4.3), and it is observed during the test. Its output rate is 32ksps, which is much faster than the digital filter

(decimator) output rate (7sps).

(4) Since the ΔΣ AD modulator contains an analog circuit, then even if there is not a fatal fault, its linearity may be degraded by parametric faults such as parasitic circuit components, which should be checked by the testing. It is assumed here that the input/output characteristics of the ΔΣ AD modulator do not have jumps (discontinuities), which is different from pipelined ADCs and SAR ADCs (Fig. 4.4).

(5) We model the input/output characteristics of the ΔΣ AD modulator including nonlinearity characteristics model with polynomials.

Let $x(t)$ be an input of the modulator and $y(t)$ be its output data stream, and then we model its input/output characteristics with the following n-th order polynomial model:

$$y(t) = a_0 + a_1 x(t) + a_2 x(t)^{2} + \cdots + a_n x(t)^{n} \tag{4-1}$$

(6) We apply a cosine wave input to the modulator as follows:

$$x(t) = A\ \cos(\omega t) \tag{4-2}$$

Here, the amplitude of $A$ is known, and the input signal frequency ω as well as the sampling clock frequency $\omega_s$ are low so that the modulator does not show high-frequency performance degradation. Then substituting Eq. (4-2) into Eq. (4-5), the modulator 1-bit output data stream is modeled by:

$$y(t) = b_0 + b_1 \cos(\omega t) + b_2 \cos(2\omega t) + \cdots + b_n \cos(n\omega t) \tag{4-3}$$

Using the coefficients $a_0, a_1, a_2, \cdots, a_n$, the coefficients $b_0, b_1, b_2, \cdots, b_n$ can be expressed as follows [10, 11]:

$$\left.\begin{aligned}
b_0 &= a_0 + \frac{1}{2}a_2A^2 + \frac{3}{2^3}a_4A^4 + \cdots \\
b_1 &= a_1A + \frac{3}{2^2}a_3A^3 + \frac{5}{2^3}a_5A^5 + \cdots \\
b_2 &= \frac{1}{2}a_2A^2 + \frac{1}{2}a_4A^4 + \frac{15}{2^5}a_6A^6 + \cdots \\
b_3 &= \frac{1}{2^2}a_3A^3 + \frac{5}{2^4}a_5A^5 + \frac{21}{2^6}a_7A^7 + \cdots \\
b_4 &= \frac{1}{2^3}a_4A^4 + \frac{1}{2^4}a_6A^6 + \frac{7}{2^5}a_8A^8 + \cdots \\
b_5 &= \frac{1}{2^4}a_5A^5 + \frac{7}{2^6}a_7A^7 + \frac{9}{2^6}a_9A^9 + \cdots \\
&\vdots \\
b_{n-1} &= \frac{a_{n-1}}{2^{n-2}}A^{n-1} \\
b_n &= \frac{a_n}{2^{n-1}}A^n
\end{aligned}\right\} \qquad (4\text{-}4)$$

(7) We perform FFT to the1-bit output data stream of the modulator and obtain $b_0, b_1, b_2, \cdots, b_n$; then we derive $a_0, a_1, a_2, \cdots, a_n$ from the relation in Eq. (4-4). Now we have the following DC input/output characteristics:

$$y(t) = a_0 + a_1 + a_2x^2 + \cdots + a_{\mathrm{n}}x^{\,\mathrm{n}} \qquad (4\text{-}5)$$

(8) Finally, we calculate the INL of the modulator from Eq. (4-5) using the end-point method or the best-straight-line method [1].

**Remark:** Here, we use the end-point method for obtain INL.

**Example:** Consider the case that the 3rd-order nonlinearity is the dominant distortion for the modulator. Then we model its input/output characteristics as follows:

$$y(t) = a_1x(t) + a_3x(t)^{\,3} \qquad (4\text{-}6)$$

Provide to the modulator a cosine wave input $x(t)$ whose amplitude $A$ is known.

$$x(t) = A\cos(\omega t)$$

Then the modulator output $y(t)$ modeled in Eq. (4-6) is given as follows:

$$\begin{aligned} y(t) &= a_1 x(t) + a_3 x(t)^3 \\ &= \left(a_1 \cdot A + \frac{3}{4} a_3 \cdot A^3\right)\cos(\omega t) + \frac{1}{4} a_3 \cdot A^3 \cos(3\omega t) \end{aligned} \tag{4-7}$$

We perform FFT to $y(t)$ and obtain its power spectrum. Then its fundamental spectrum power is given as follows:

$$b_1 = a_1 \cdot A + \frac{3}{4} a_3 \cdot A^3 \tag{4-8}$$

Its third harmonic spectrum power is expressed as follows:

$$b_1 = \frac{1}{4} a_3 \cdot A^3 \tag{4-9}$$

We can estimate the polynomial coefficients $a_1$, $a_3$ in Eq. (4-6) from $b_1$, $b_3$, and $A$, using the relationship among $a_1$, $a_3$, $b_1$, $b_3$, and $A$ in Eqs. (4-8), (4-9). Now we can estimate the AD ΔΣ modulator characteristic given by Eq. (4-6) and then calculate the overall ADC INL with the end-point method.

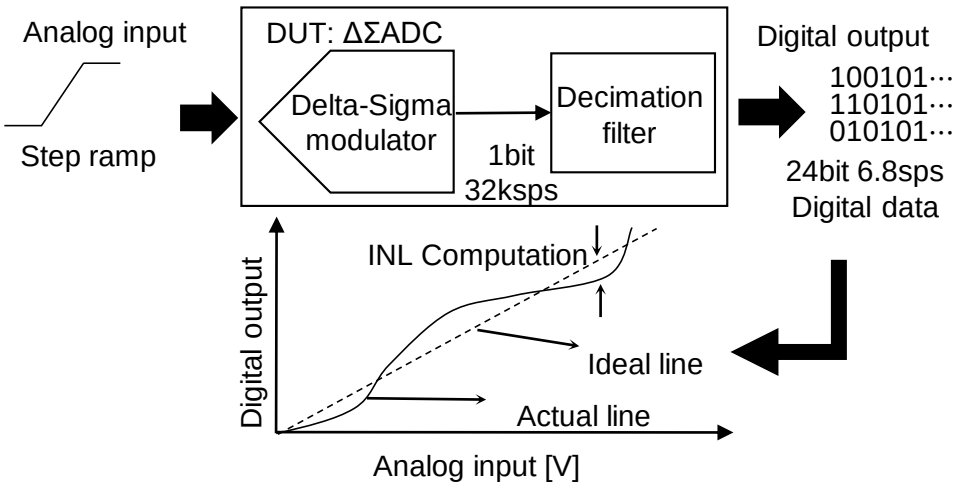



Fig. 4.2 All code testing for INL testing.

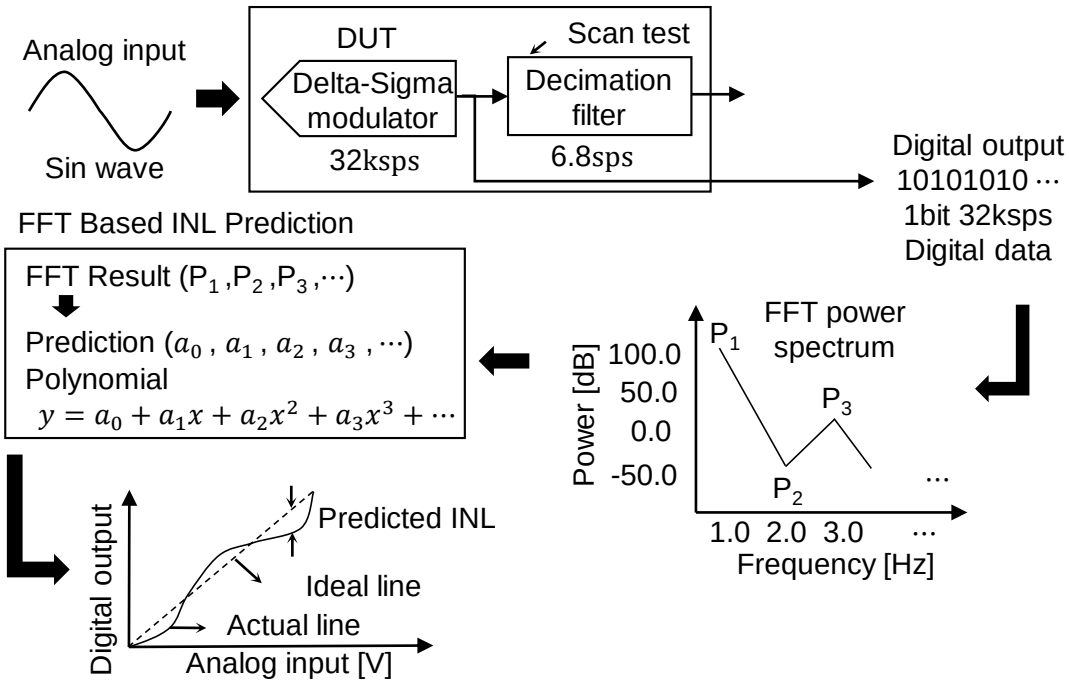



Fig. 4.3 Proposed FFT-based INL prediction method.

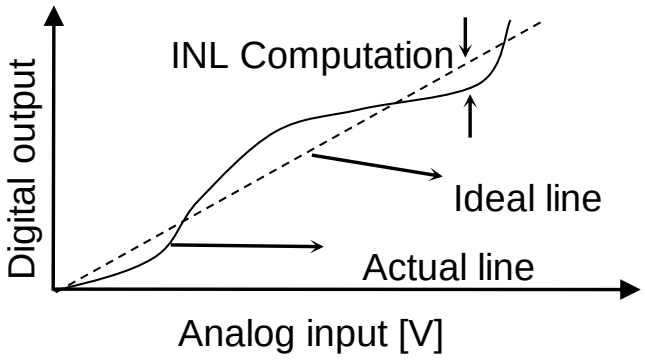



Fig. 4.4 Input/output characteristics of the ΔΣAD modulator without jumps.

## 4.5 Simulation verification of proposed integral linearity test for ΔΣ ADC modulator

### 4.5.1 Simulation condition

Section 4.5 shows simulation verifications of the proposed method in Section 4.3, in the following cases:

(i) Discrete-time 1st-order and 2nd-order modulators.

(ii) 3rd-order and 5th-oder nonlinearities.

(iii) Several nonlinearity strength variations.

(iv) Several input cosine wave amplitudes.

(v) Several cases for the number of acquired 1-bit data stream of the modulator output.

Fig. 4.5 shows our simulation model of the 1st-order ΔΣ AD modulator with nonlinearities.

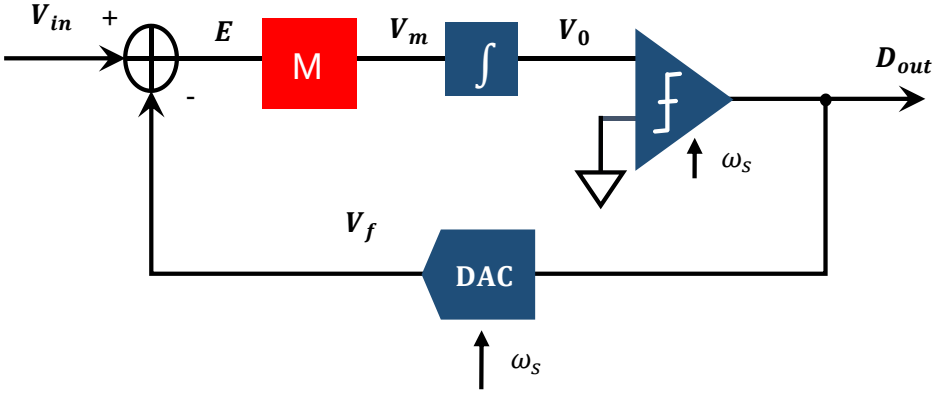



Fig. 4.5 Simulation model of the 1st - order ΔΣAD modulator with nonlinearity.

Here

$$\mathrm{E(n)} = \mathrm{V_{in}(n)} - \mathrm{V_f(n)}$$

$$\mathrm{V_o(n)} = \mathrm{V_o(n-1)} + \mathrm{V_m(n)}$$

$$\text{If} \quad \mathrm{V_o(n)} \geq 0, \text{then}$$

$$\mathrm{D_{out}(n+1)} = 1;\ \mathrm{V_f(n+1)} = 1$$

$$\text{Else}\ \ \mathrm{D_{out}(n+1)} = 0;\ \mathrm{V_f(n+1)} = -1$$

The block M models the modulator nonlinearity and its nonlinearity strength can be controlled by the parameter $k$. Also notice that the block diagram in Fig. 4.5 is for system level simulation with MATLAB; in the actual circuit, the DAC output ($V_f$) is $V_{ref}$ for $D_{out}$ =1, or – $V_{ref}$ for $D_{out}$ = 0, and the range of the ADC input ($V_{in}$) is from -$V_{ref}$ to $V_{ref}$.

In case that 3rd-order nonlinearity is dominant, we use

$$\mathrm{V_m(n)} = \mathrm{E(n)} - k * \mathrm{E(n)}^3 \quad (k > 0) \tag{4-10}$$

In case that 5th-order nonlinearity is dominant, we use

$$\mathrm{V_m(n)} = \mathrm{E(n)} - k * \mathrm{E(n)}^5 \quad (k > 0) \tag{4-11}$$

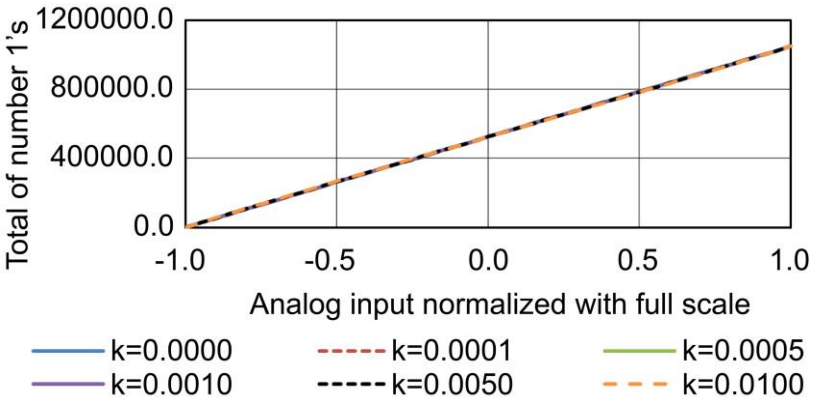
In subsections 4.5.2 - 4.5.6, we consider the 3rd - order harmonics is dominant and use Eq. (4-10), whereas in subsection 4.5.6 we consider also 5th-order harmonics is dominant and use Eq. (4-11).

### 4.5.2 DC input-output characteristic with cure fitting

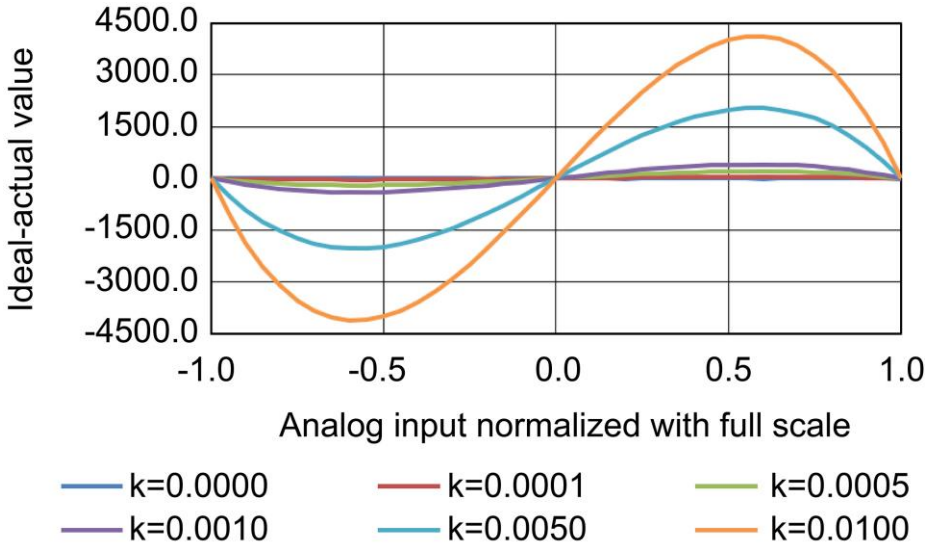
In our first simulation, we apply a DC input to the 1st-order modulator input $V_{in}$ in Fig. 4.5, from -1 to 1 with 0.05 step and obtain its input/output characteristics as a reference, even though it takes quite a large number of AD modulator sampling points. The number of 1's ($D_{out}$ = 1) for each DC value is obtained using $2^{20}$ data for a given DC input: the input DC value is changed with 0.05 step so that the total sampling number to obtain the whole input/output is enormous. The value of $k$ representing the strength of the nonlinearity is varied as 0.0000, 0.0001, 0.0005, 0.0010, 0.0050, 0.0100, and the number of 1's at $D_{out}$ is plotted in Fig. 4.6, which is the DC input/output characteristics and the INL of the modulator in Fig. 4.5.

The input/output characteristics in Fig. 4.5 are polynomial approximated by the following formula:

$$y = a_0 + a_1 \cdot x + a_2 \cdot x^2 + a_3 \cdot x^3 \qquad (4\text{-}12)$$




(a) Modulator output number of 1's




(b) Difference between the ideal and actual modulator output numbers of 1's.

Fig. 4.6 Simulation results of the DC input/output characteristics and the INL of the ΔΣAD modulator in Fig. 4.5 when the number of the modulator output is $2^{20}$.

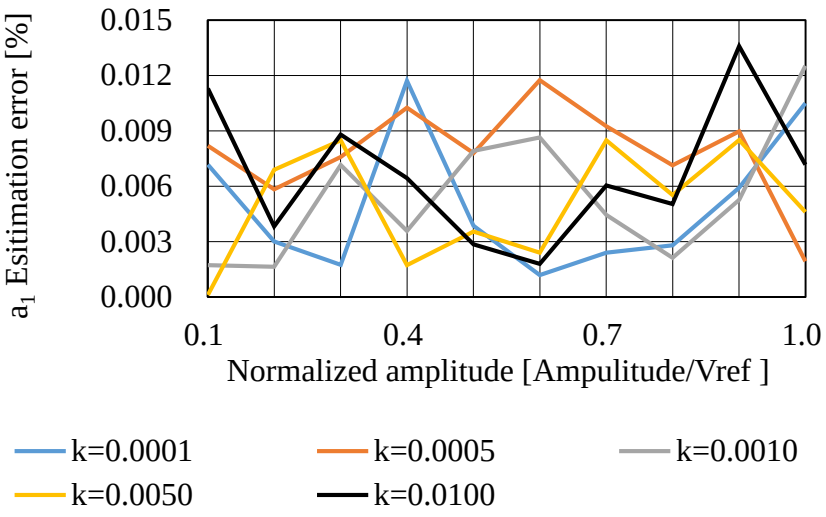
Table 4.1 shows the values $a_1$, $a_3$ obtained from the simulation results in Fig. 4.6 using the curve fitting for each $k$, $a_0$, and $a_2$, are relatively very small due to the nonlinearity model usage of Eq. (4-10). So they can be ignored and are not written in Table 4.1. We see the followings from Table 4.1:

(1) As the value of $k$ increases, the values of $a_1$ slightly decreases.

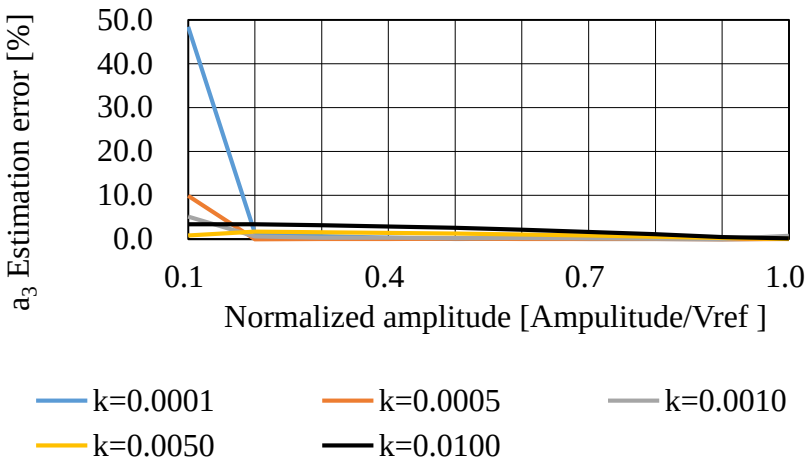
(2) As the value of $k$ increases, the value of $a_3$ increases.

Table 4.1 Estimated coefficient values in the polynomial model of the ΔΣ AD modulator DC input/input characteristics.

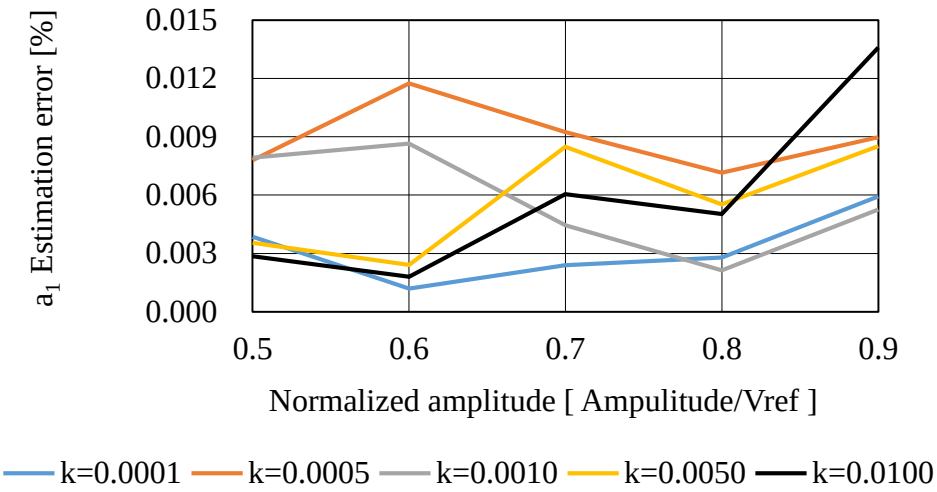
| $k$ | $a_1$ | $a_3$ |
|---|---|---|
| 0.0001 | 524180 | 104.84 |
| 0.0005 | 523760 | 524.48 |
| 0.0010 | 523240 | 1050.50 |
| 0.0050 | 519000 | 5282.50 |
| 0.0100 | 513610 | 10643.00 |



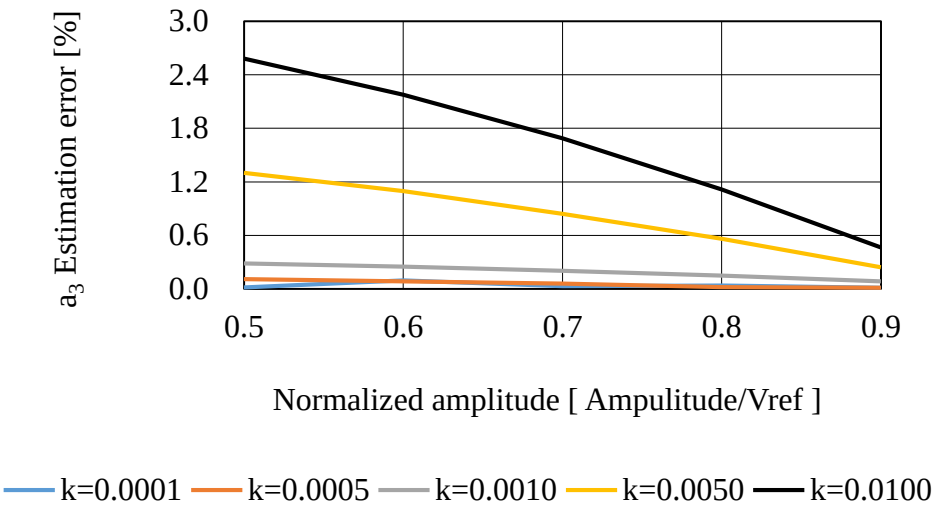

(a) $a_1$ estimation error (input: 0.1 ~ 1.0)




(b) $a_3$ estimation error (input: 0.1 ~ 1.0)




(c) $a_1$ estimation error (input: 0.5 ~ 0.9)




(d) $a_3$ estimation error (Input: 0.5 ~ 0.9)

Fig. 4.7 Estimation errors of the polynomial coefficients obtained from the 1st-order modulator output power spectrum.

### 4.5.3 Cosine wave input and output power spectrum

Next, we consider to provide a cosine wave to the AD modulator as $V_{in}$ (Eq. (4-2), Fig. 4.3), and obtain its 1-bit output stream of $2^{20}$ data. Then we perform FFT for the 1-bit output data stream and obtain its power spectrum; the fundamental wave power $P_1$ and the third harmonic power $P_3$ (Fig. 4.7). Here $\omega_{in}/\omega_s = 1/2^{20}$, $\omega_{in}$ is an input angular frequency and $\omega_s$ is a sampling angular frequency.

As the number of the acquired modulator output data is large, the estimation accuracy for $a_1$, $a_3$ improves; we found that $2^{20}$ is a reasonable compromise between testing time and accuracy for both the1$^{st}$-order and 2$^{nd}$-order modulators.

### 4.5.4 Estimation of Polynomial Coefficients with Proposed Method

Polynomial modeling is performed for the DC input/output characteristics of the ΔΣ AD modulator, based on Eq. (4-6). Then we estimate the values of $a_1, a_3$ from $P_1$ and $P_3$ obtained in Fig. 8, using Eqs. (4-8) and (4-9). Fig. 4.7 shows the errors of $a_1, a_3$ between these estimates and the ones in Table 4.1.

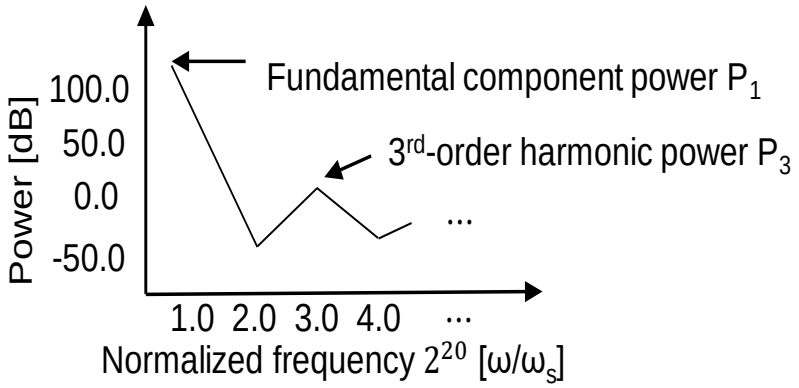



Fig. 4.8 Simulation result of the ΔΣ modulator output power spectrum obtained by FFT for a cosine wave input.

### 4.5.5 2nd-order Modulator Case

A 2nd-order modulator in Fig. 4.9 with $2^{20}$ samples is also simulated with the same method. Fig. 4.10 shows the estimation errors of the fundamental and 3rd harmonics for the input with the amplitude of 0.5 to 0.9 from the 2nd-order modulator output power spectrum. We see the following from Figs. 4.8 and 4.10:

(1) Estimation error for $a_1$ is small for all the input amplitude *A*.

(2) When the input amplitude *A* increases to 0.9, then the estimation error for $a_3$ is reduced. Notice that the input amplitude *A* is controllable during testing time.

(3) Notice that if the number of data is reduced compared to $2^{20}$, the

estimation error becomes larger than the one as shown in Figs. 4.6 and 4.7.

(4) By comparing the 1st-order modulator with the 2nd-order modulator, the estimation errors of the both models are small. So we expect that this method is applicable for testing also high-order modulators.

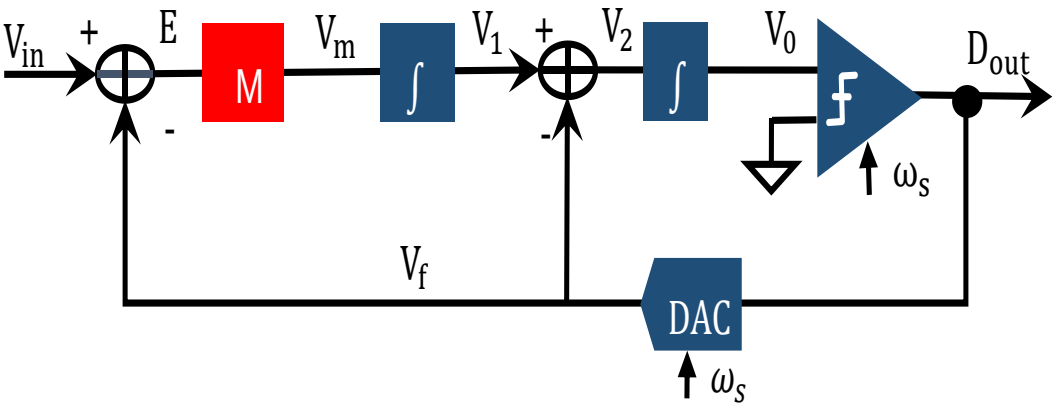



Fig. 4.9 Simulation model of the 2nd -order ΔΣAD modulator with nonlinearity.

Here:

$$\mathrm{E(n)} = \mathrm{V_{in}(n)} - \mathrm{V_f(n)}$$

$$V_1(n) = V_1(n-1) + V_m(n)$$

$$V_2(n) = V_1(n) - V_f(n)$$

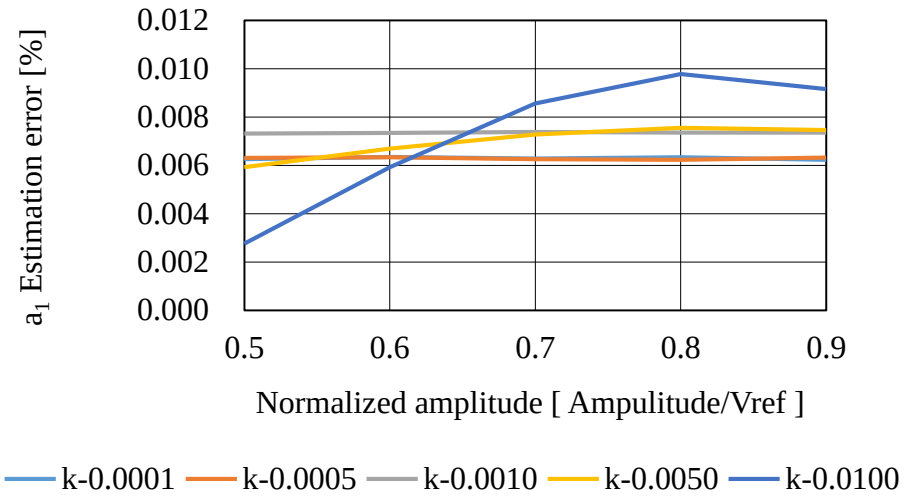
$$\mathrm{V_o(n)} = \mathrm{V_o(n-1)} + \mathrm{V_2(n)}$$

$$\text{If} \quad \mathrm{V_o(n)} \geq 0, \text{then}$$

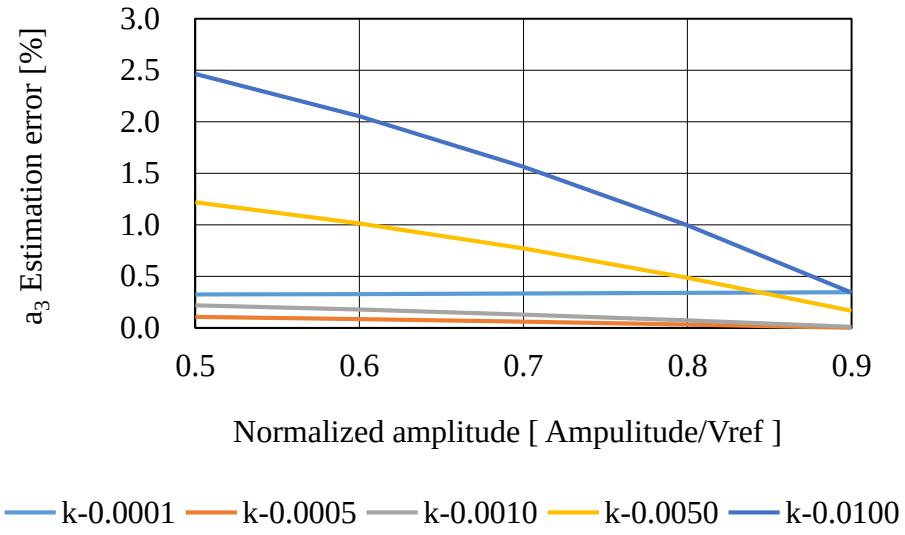
$$\mathrm{D_{out}(n+1)} = 1;\ \mathrm{V_f(n+1)} = 1$$

$$\text{Else} \quad \mathrm{D_{out}(n+1)} = 0;\ \mathrm{V_f(n+1)} = -1.$$

The block M models the modulator nonlinearity and its nonlinearity strength can be controlled by the parameter $k$. In case that 3rd-order or 5th-order nonlinearity is dominant, we use Eq. (4-10) or Eq. (4-11) respectively.




(a) $a_1$ estimation error (Input: 0.5 ~ 0.9)




(b) $a_3$ estimation error (input: 0.5 ~ 0.9)

Fig. 4.10 Estimation errors of the polynomial coefficients obtained from the 2nd - order modulator output power spectrum.

### 4.5.6 Estimation of INL with Proposed Method

In this subsection, the INL is estimated based on our prosed FFT method and compared with the reference INL obtained by the curve fitting method using simulations. We found that the amplitude 0.9 is the best value for the accurate INL estimation, and notice that during test, the input to the ΔΣADC under test can be controlled so that the input of 0.9 cos(ωt) can be provided. Table 4.2 shows the 3rd-order and 5th-order harmonics comparison. Fig. 4.11 shows their comparison for the amplitude of 0.9 when the 3rd-order harmonics is dominant (Eq. (4-10) is used), while Fig. 4.12 is the one when the 5th-order harmonics is dominant (Eq. (4-11) is used). The vertical axes in Figs. 4.11 (a), (b) and 4.12 (a), (b) show errors of the modulator output 1's number for $2^{20}$ data when 1LB is considered as $1/2^{20}$. We see in Figs. 4.11 (c) and 4.12 (c) that estimated INL errors with our proposed method are sufficiently small.

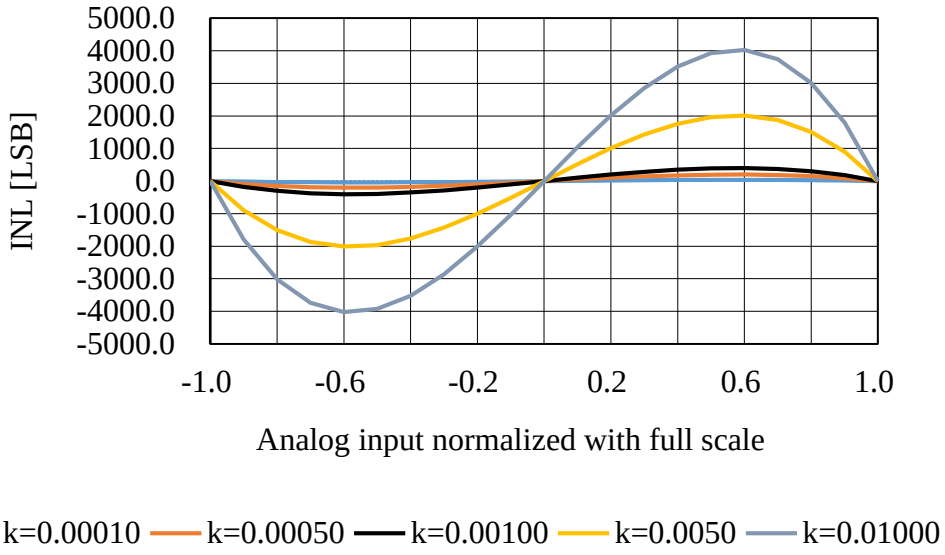
Fig. 4.13 shows INL errors as a function of the number of the modulator output data for the amplitude of 0.9 and k=0.0005; $2^{14}$, $2^{16}$, $2^{18}$, $2^{20}$ and $2^{22}$. We see that the number of the data is larger, the error is smaller.

Table 4.2 3rd and 5th harmonics estimation maximum error.

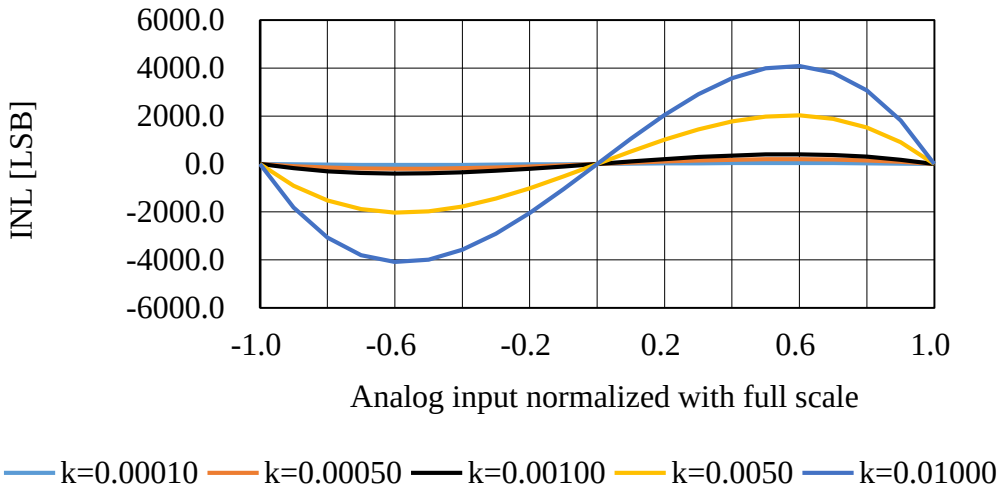
| *k* | 3rd harmonic estimation error [%] | 5th harmonic estimation error [%] |
|---|---|---|
| 0.0001 | 0.0187 | 0.1421 |
| 0.0005 | 0.0213 | 0.1037 |
| 0.0010 | 0.0784 | 0.0142 |
| 0.0050 | 0.2342 | 0.0092 |
| 0.0100 | 0.4516 | 0.0276 |

## 4.6 Experimental Verification

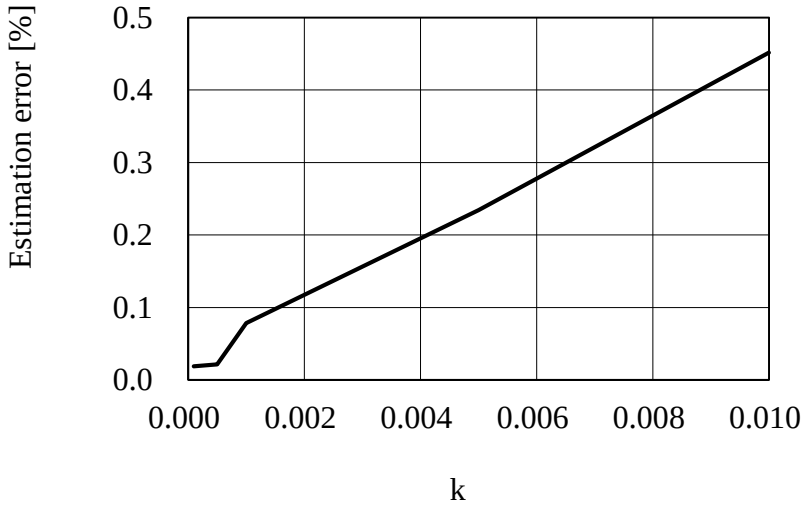
Our research collaborators of ROHM Semiconductor Ltd. have performed experiments with a real ΔΣ ADC chip using the proposed algorithm. Our target INL test accuracy is within ± 1ppm, so that requirements for the input signal source are that THD < -120dB and SN > 130dB and synchronization between the signal source and the DUT (Device Under Test) of the ΔΣADC in Fig. 4.14. Then we have developed a precise arbitrary waveform generator (AWG) whose performance is shown in Fig. 4.15. and the NI PXI system in Fig. 4.16. is used and test environment in Fig. 4.17. The modulator output FFT results are obtained in Fig. 4.18, and the INL prediction is shown in Fig. 19. These results show that our proposed method can estimate the INL at ppm level.




(a) INL obtained by our proposed FFT method

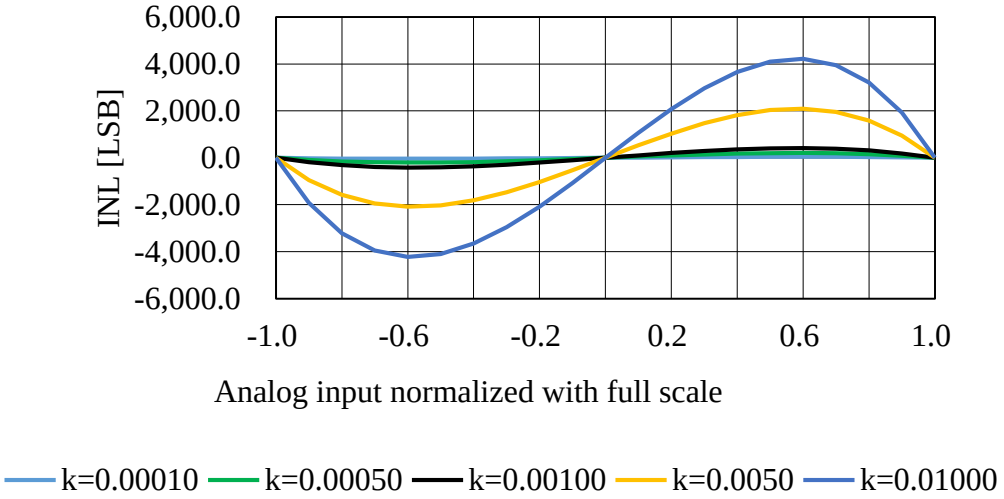



(b) INL obtained by the curve fitting method

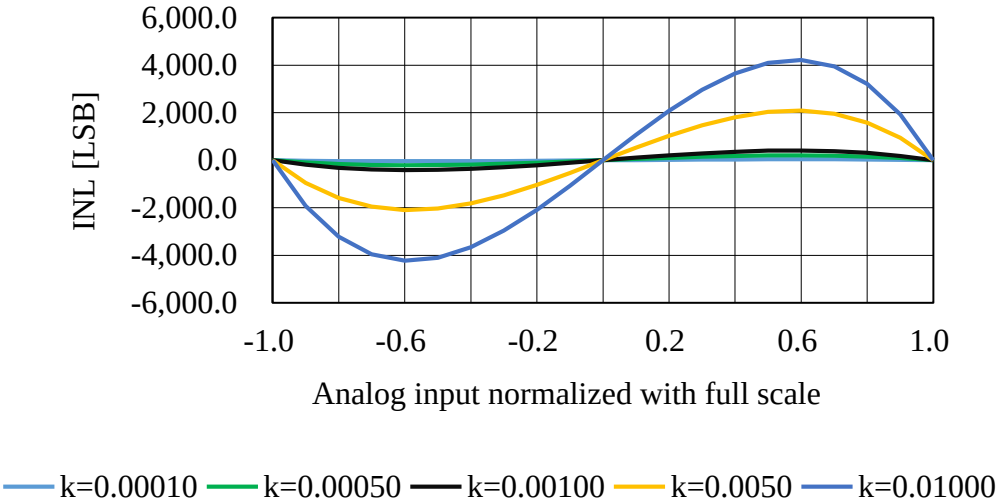



(c) INL error between the FFT and curve fitting methods

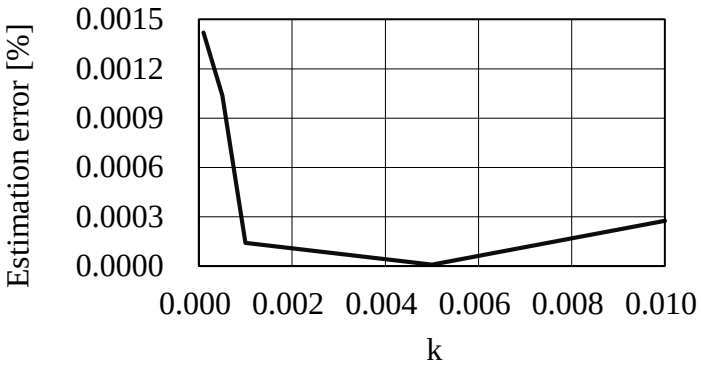
Fig. 4.11 INL comparison between the FFT and curve fitting methods when the 3rd-order harmonics is dominant.




(a) INL obtained by our proposed FFT method

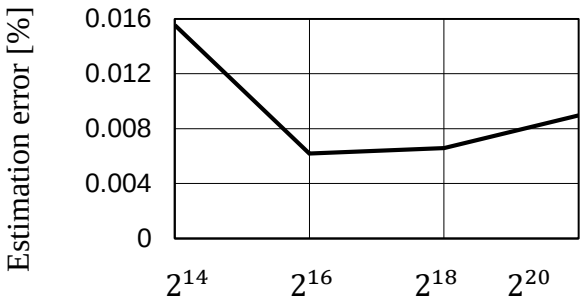



(b) INL obtained by the curve fitting method

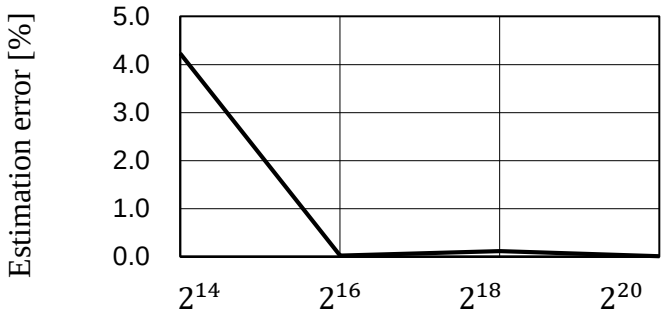



(c) INL error between the FFT and curve fitting methods

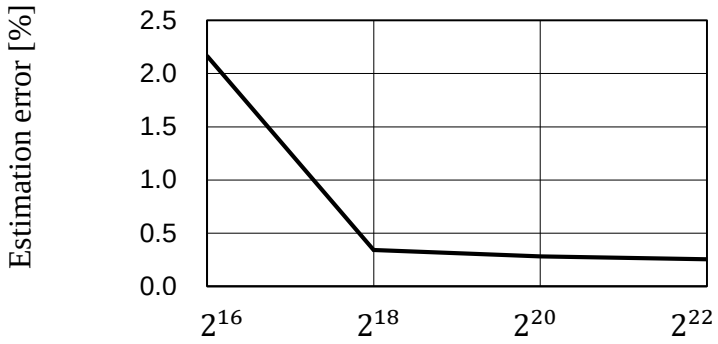
Fig. 4.12 INL comparison between the FFT and curve fitting methods when the $5^{th}$ - order harmonics is dominant.




(a) $a_1$ estimation error



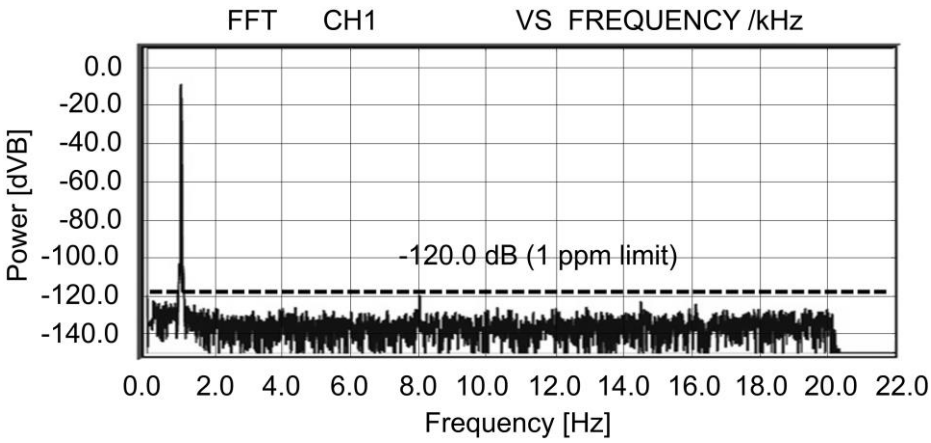

(b) $a_3$ estimation error




Number of modulator output data

(c) $a_5$ estimation error

Fig. 4.13 Number of the modulator output data and estimation errors for $a_1$, $a_3$, $a_5$ with the proposed FFT method.

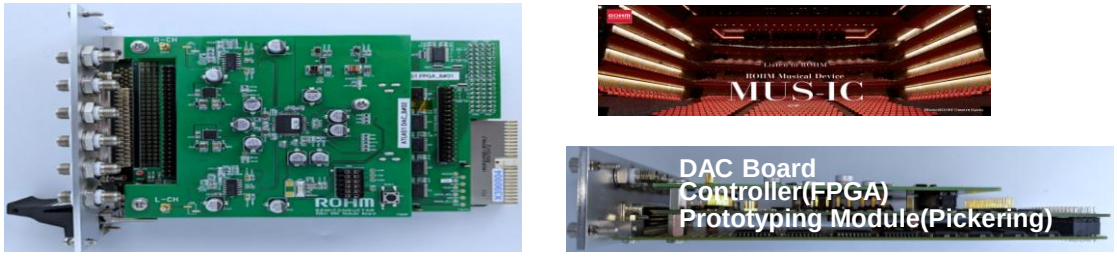



Output: 1kHz 44.1ksps

THD: 122dB (~5$^{th}$-order harmonics)

SN: 131dB (Filter:20kHz LPF)

Fig. 4.14 Signal from our developed AWG.




(a) PXI compatible module (b) ROHM 32bit audio DAC

Fig. 4.15 Development of precise signal generation AWG.

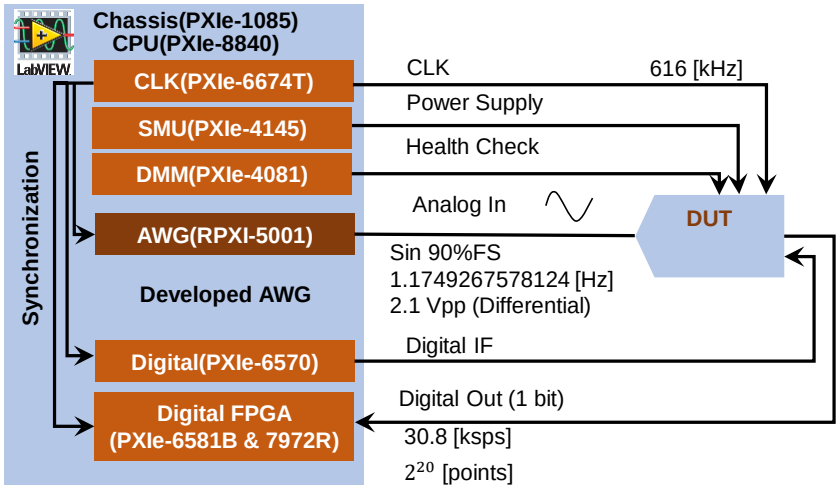
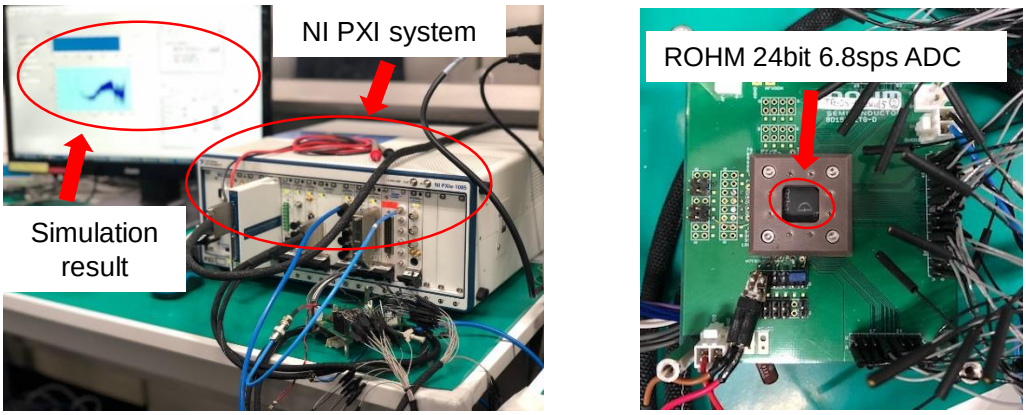
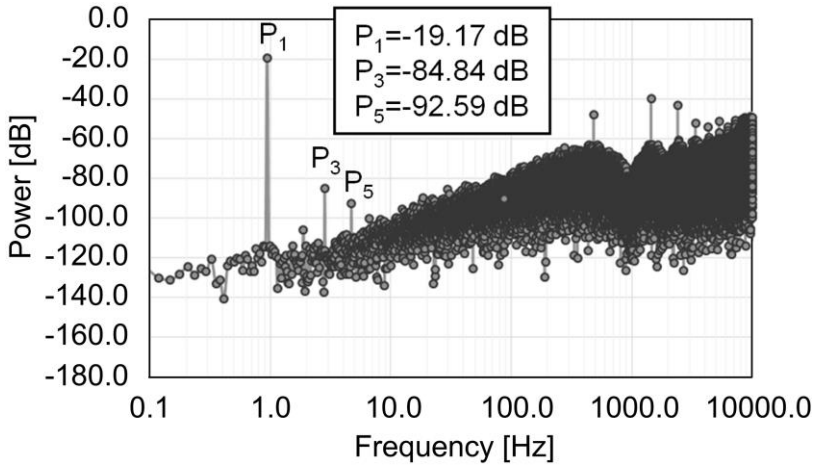



Fig. 4.16 Use of NI PXI system for experiment.




(a) PXI setup (b) DUT board

Fig. 4.17 Test environment.




Input amplitude A= 2.252Vpp (differential)

Fig. 4.18 Experimental result of the modulator output FFT.

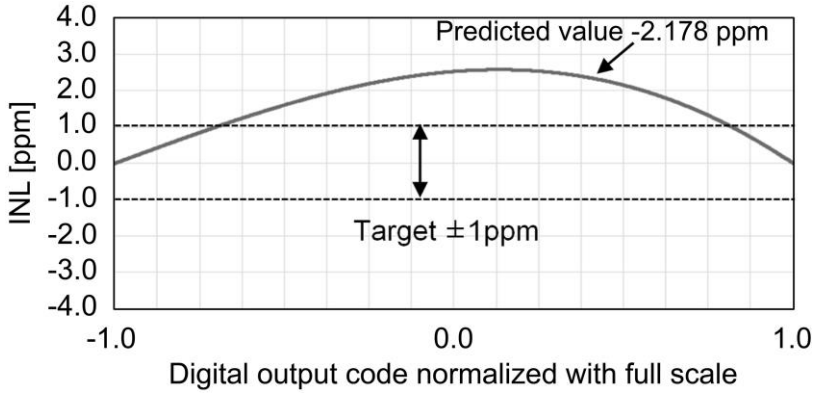



Fig. 4.19 Obtained INL prediction with the proposed method.

## 4.7 Discussions

(i) Integral nonlinearity test time estimation:

Suppose that the ΔΣAD modulator operates with 32ksps and the required number of data for INL test is $2^{20}$. Then the required testing time is 32 seconds. If 32 chips are tested in parallel, the equivalent test time per chip is 1 second; this test time may be acceptable for industry applications.

The following are calculation equations:

a) The INL testing time with the direct method for a 7-sps 24-bit ΔΣADC is calculated by

$$\text{Test Time} = 2^{24} * \frac{1}{7} * n \ [\text{sec}] = 666[\text{h}] * n \quad \text{Here, } n = \frac{\text{Samples}}{\text{Code}}$$

In case n=4, its testing time is 111 days.

b) On the other hand, when the proposed method is used, the testing time is given by

$$\text{Test Time} = 2^{20} * \frac{1}{32{,}000} = 32[\text{sec}]$$

(ii) One might claim that if the gain of the operational amplifier inside the modulator is not high enough, the input/output characteristics of the ΔΣ AD modulator can have jumps and it is not continuous [4]. However, our target ΔΣ ADC does not have the jumps with some circuit techniques.

(iii) For the direct INL method, a precise DC signal generator with more than 24-bit resolution is required for the modulator input. However, for the proposed method, a low distortion signal generator and a low-pass filter such as [12] are enough. Recently 32-bit ΔΣ ADCs are commercially announced and there would not be a DC signal generator for their INL testing with the direct method; an ultra-high-precision DC signal source is difficult to realize.

(iv) The proposed method can be applied for high-order modulators, continuous-time modulators and multi-bit modulators; this is under investigation.

(v) The proposed method can be considered as solving so-called an inverse problem. The modulator nonlinearity is modeled as a polynomial and its coefficients are estimated by the FFT method; the modulator INL is indirectly measured by its output power spectrum.

## 4.8 Summary

We have proposed a short-time high-accuracy integral linearity test method/algorithm of the high-resolution low-sampling-rate ΔΣADC for mass production. We have conducted its modeling and simulation as well as experimental verification. For the next step, we will take higher-order distortions into account, and apply the proposed method to higher-order modulators. We will also perform further experiments with real ΔΣADC chips and verify the test time in the ATE environment.

# References


[1] G. Robert, F. Taenzler, M. Burns, An Introduction to Mixed-Signal IC Test & Measurement, 2nd Edition, London, UK (2012).

[2] Y. Sasaki, K. Machida, R. Aoki, S. Katayama, T. Nakatani, J. Wang, K. Sato, T. Ishida, T. Okamoto, T. Ichikawa, A. Kuwana, K. Hatayama, H. Kobayashi, “Accurate and Fast Testing Technique of Operational Amplifier DC Offset Voltage in μV-order by DC-AC Conversion”, 3rd International Test Conference in Asia, Tokyo, Japan (Sept. 2019).

[3] H. Kobayashi, K. Kobayashi, H. Sakayori, Y. Kimura, “ADC Standard and Testing in Japanese Industry", Computer Standards & Interfaces, Elsevier Publishers, vol. 23, pp.57-64 (Mar. 2001).

[4] R. Schreier, S. Pavan, G. C. Temes, Understanding ΔΣ Data Converters, 2nd Edition, New Jersey (2017).

[5] J. M. Rosa, ΔΣ Converters: Practical Design Guide, 2nd Edition, New Jersey (2018).

[6] J. Kojima, N. Kushita, M. Murakami, H. Kobayashi, “Linearity Enhancement Algorithms of Multi-bit ΔΣ DA Converter-DWA, Self-Calibration and Their Combination”, Journal of Technology and Social Science, Vol.2, No.2, pp.15-28 (Mar. 2018).

[7] J.-L. Wei, N. Kushita, H. Kobayashi, “Limit Cycle Suppression Technique Using Random Signal In ΔΣ DA Modulator”, Proceedings of IEEE 14th International Conference on Solid-State and Integrated Circuit Technology: ICSICT-2018, Qingdao, China (Oct. 2018).

[8] M. Murakami, H. Kobayashi, S. N. B. Mohyar, O. Kobayashi, T. Miki, J. Kojima, “I-Q Signal Generation Techniques for Communication IC Testing and ATE Systems”, Proceedings of IEEE International Test Conference, Fort Worth, TX (Nov. 2016).

[9] F. Maloberti, Data Converters, New York, USA (2007).

[10] P. Wambacq, W. Sansen Distortion Analysis of Analog Integrated Circuits, Boston, USA (2013).

[11] F. Abe, Y. Kobayashi, K. Sawada, K. Kato, O. Kobayashi, Haruo Kobayashi, “Low-Distortion Signal Generation for ADC Testing”, IEEE International Test Conference: ITC2014, Seattle, WA (Oct. 2014).

[12] T. Komuro, S. Sobukawa, H. Sakayori, M. Kono, H. Kobayashi, “Total Harmonic

Distortion Measurement System for Electronic Devices up to 100 MHz with Remarkable Sensitivity", IEEE Trans. on Instrumentation and Measurement, Vol. 56, No. 6, pp. 2360-2368 (Dec. 2007).

# Chapter 5

# CONCLUSIONS AND FUTURE WORK

## 5.1 Conclusions

This part starts with a brief introduction to the concept of ADCs and the evaluation criteria for ADCs, giving the reader a preliminary understanding of ADCs and the criteria for judging them. This is followed by a brief comparison of the advantages and disadvantages of different ADC types in terms of structure, conversion principle, adoption speed, accuracy, power and application scenarios. In today's rapidly developing electronics industry, there are certain application areas where traditional naive ADCs are no longer adequate, but ΔΣ ADCs have their own unique performance advantages. For example, they play an important role in applications where high precision AD conversion is required.

For ΔΣ ADCs in high-precision conversion, the quantizer requirements are very high. In analog circuits, it is still difficult to improve the quantizer, but in digital circuits this can be easily implemented. Based on the advantages and disadvantages of both analog and digital, we decided to start with the design of the circuit from the digital side of the signal. Various factors in the AD conversion process have an impact on the accuracy of the conversion due to the quantizer. Therefore, we proposed a method to control the output of the signal by replacing the grounded port in the quantizer section with access to a random signal, which effectively suppresses the generation of limit loops, thus effectively improving the performance of the conversion. The spectrograms shown in the dissertation can clearly show that

by comparing the spectrograms before and after the addition of the random signal, the addition of the random signal can effectively reduce the noise. A comparison of the size of the added random signal was also made, and the best SFDR values were obtained for random signals in the range of -2 ~ +2.

Similarly, ΔΣ ADCs can be well used in the field of sensor interface circuits. In this paper, a test method for high precision INL using FFT has been developed using ΔΣ ADCs. The method starts by obtaining the DC output curve from the input DC signal and then approximating the curve equation to obtain the fundamental and harmonic coefficients. The input signal is then changed to a cosine signal to obtain an output FFT spectrum, from which the fundamental and harmonic values are obtained. Finally, we have compared the third and fifth harmonic simulations by calculating the obtained inferred values and the values obtained from the FFT, as well as combining the results of the hardware tests by our research collaborators at ROHM Semiconductor Ltd., proving that our proposed test method is valid and can also significantly reduce the test time.

## 5.2 Future work

There are still areas of deficiency in this design that need further refinement and improvement for subsequent improvements.

What is the limit value of the input random signal that can be reached during a high precision AD conversion? Whether the method has the same effect on multi-bit ΔΣ ADCs.

Whether the method can be applied in embedded ΔΣ ADC structures such as ΔΣ-pipeline, SAR-ΔΣ structures in test applications. In these structures, it is possible to increase the AD conversion rate and thus save test

time.

# PART 2

# Chapter 1

# INTRODUCTION

## 1.1 Research Background and Motivation

With the advancement of semiconductor process technology and the development of floating-point processor design technology, floating-point processors have gone through different stages such as digital co-processors, single floating-point processing unit integrated in microprocessors and multiple floating-point processing units integrated [1-3]. To meet the needs of everyday life, integrated circuit (IC) chips such as high-performance microprocessors based on integrated circuit technology are increasingly becoming a catalyst for social development. At the same time, the demand for performance in various electronic products has led to the development of integrated circuit manufacturing technologies to meet design requirements [4,5]. It is the continuous development of IC manufacturing processes that has led to effective improvements in the speed, power consumption and chip area of microprocessors, which are at the heart of information processing. In microprocessors, floating-point numbers, with their own particular ability to perform accurately and over a wide range of data, have been widely studied as a fundamental requirement.

However, the structure design of the floating-point operator units is more complicated than that of the fixed-point operator units, and it consumes more hardware resources, so at the early stage of IC development, many ICs did not contain special floating-point units, so for floating-point operations most computers use fixed-point units through software to achieve, although

this method makes the structure design of the IC simpler and reduces the consumption of hardware resources. Although this approach simplifies the design of the IC structure and reduces the consumption of hardware resources, the speed of the floating-point calculation is significantly reduced, so it is not feasible to use software to calculate floating-point numbers where speed is required. In addition to this, there are also manufacturing process barriers. Previous processes made it difficult to implement floating-point units in hardware, but now we know that IC process features are getting smaller and smaller, and that System-on-Chip (SoCs) can be implemented [6,7], so it is now possible to embed a separate floating-point unit inside the hardware [8].

In the field of digital signal processing, a Digital Signal Processor (DSP) is a microprocessor dedicated to digital signal processing. With more arithmetic power than general purpose microprocessors, DSPs are used in a wide range of applications including radar, sonar, communications, multimedia, biomedicine, sensing networks, automotive and aircraft control, and precision-guided weapons, and have profoundly influenced the development of modern society [9-11]. Current digital signal processing systems have been changing from fixed-point arithmetic to floating-point arithmetic.

In summary, the difficulty with floating-point calculations lies in the calculation of the mantissa. The aim of this paper is to improve the performance of basic floating-point arithmetic calculations by using Taylor-series expansions, so as to use the least number of Taylor-series expansions to achieve high-precision design of floating-point mantissa.

## 1.2 Organization

The first chapter presents a brief description of the technical difficulties encountered in the design of IC chips and DSPs for floating-point calculations. Finally, the purpose of this dissertation research is explained. In Chapter 2, the format of floating-point representation, the different floating-point precision, and the different approaches to floating-point calculations are introduced. In Chapters 3, 4, 5 and 6, we investigated the use of the proposed Taylor-series expansion to compute floating-point mantissa. In the proposed Taylor-series expansion calculations region uniform division, region non-uniform division, and region conversion division are used in the calculation of different floating-point number mantissa, respectively. In Chapter 7, we summarize this part and future work.

# Reference


[1] T. Sukemura, "FR 500 VLIW-Architecture High-Performance Embedded Microprocessor", Fujitsu Scientific and Technical Journal, Vol. 36, No. 1, pp. 31-38 (Jan. 2000).

[2] J. G. Tong, I. D. L. Anderson and M. A. S. Khalid, "Soft-Core Processors for Embedded Systems", International Conference on Microelectronics, Dhahran, Saudi Arabia (Jun. 2006).

[3] J. Takala, "General-Purpose DSP Processors", Handbook of Signal Processing Systems. Springer, Boston, MA (2010).

[4] S. Galal and M. Horowitz, "Energy-Efficient Floating-point Unit Design", IEEE Transactions on Computers, vol. 60, no. 7, pp. 913-922 (Jul. 2011).

[5] H. Oh, S. M. Mueller, C. Jacobi, K. D. Tran, et. al., "A Fully-Pipelined Single-Precision Floating-point Unit in The Synergistic Processor Element of a CELL Processor", IEEE Journal of Solid-State Circuits, vol. 41, no. 4, pp. 759-771 (Apr. 2006).

[6] K. K. Anumandla, R. Peesapati, et. al., "SoC Based Floating-point Implementation of Differential Evolution Algorithm Using FPGA", Design Automation for Embedded Systems, vol. 16, no. 4, pp. 221-240 (Jan. 2012).

[7] N. Gajjar, N. M. Devahsrayee and K. S. Dasgupta, "Scalable LEON 3 Based SoC for Multiple Floating-point Operations", Nirma University International Conference on Engineering, Ahmedabad, India (Dec. 2011).

[8] L. Benini, E. Flamand, D. Fuin and D. Melpignano, "P2012: Building an Ecosystem for a Scalable, Modular And High-Efficiency Embedded Computing Accelerator", Design, Automation & Test in Europe Conference & Exhibition, Dresden, Germany (Mar. 2012).

[9] R. Bhargava, L. K. John, B. L. Evans and R. Radhakrishnan, "Evaluating MMX Technology Using DSP And Multimedia Applications", 31st Annual ACM/IEEE International Symposium on Microarchitecture, Dallas, TX (Dec. 1998).

[10] E. B. Sudderth and W. T. Freeman, "Signal and Image Processing with Belief Propagation [DSP Applications]," IEEE Signal Processing Magazine, vol. 25, no. 2, pp. 114-141, (Mar. 2008).

[11] P. TANG, and D-X. LU, "Application of DSP and Software Radio in Wireless Communication," Communications Technology, Vol. 43. No. 6, pp. 224-226 (Jun. 2010).

# Chapter 2

# INTRODUCTION TO FLOATING-POINT REPRESENTATION AND BASIC OPERATIONS

## 2.1 Floating-point representation format and classification

In the process of computer numerical calculations, fixed-point numbers represent a limited range of positive and negative integers centered on 0. As the decimal point position of fixed-point numbers is relatively fixed, it is not suitable for representing numbers with a large dynamic range, and equally unsuitable for representing numbers with a small range, so it has a major drawback in terms of numerical accuracy. In contrast to the fixed-point format, the order of a floating-point number can be varied within a certain range, while the position of the decimal point varies according to the specific calculated value. This allows a wider range of values to be expressed more flexibly, and so the floating-point representation is widely used. The Institute of Electrical and Electronics Engineers (IEEE) developed the IEEE-754 standard for floating-point numbers in 1985, and this standard is now widely used [1,2]. The IEEE-754 standard covers the following main areas:

(1) Basic and extended floating-point formats.

(2) Operations such as addition, subtraction, multiplication and division.

(3) Conversions between integer and floating-point formats.

(4) Conversions between different floating-point formats.

(5) Conversions between floating-point formats and decimal numbers.

(6) Floating-point exceptions and how they are handled.

Floating-point operations are more widely used than fixed-point operations, but most FPGA synthesis tools do not support floating-point operations due to the more complex circuitry of floating-point devices, which requires the design of floating-point devices to implement basic floating-point operations. Here we introduce the basic concepts and international standards of floating-point arithmetic, and then describe the basic theory of floating-point arithmetic, including the IEEE-754 binary floating-point standard, the floating-point memory format standard, and normalized shifts.

In general, IEEE-754 standard floating-point numbers consist of sign bit, exponent part and mantissa part. The data in a floating-point number consists of two numbers, 0 and 1, with the base implied as 2. The format is shown in Fig. 2.1.


Fig. 2.1 Format of floating-point number

The sign bit (S) represents whether the floating-point number is positive or negative and occupies only 1 bit, with 0 representing a positive number and 1 representing a negative number. The exponent (*E*) part is represented as an integer, that is to satisfy the valid data part of the exponent, and to represent the positive or negative exponent. The IEEE-754 standard for this type of exponent specifies that the exponent part is the sum of the two numbers using the actual exponent plus an offset for the actual storage, that

is $e = E$ - Bias, where $E$ is called the exponent bit, and bias represents the offset. The mantissa (M) part is a positive number and is expressed as a fixed-point decimal, the first bit before the mantissa decimal of a specific floating-point number is always "1"; therefore, when storing the valid mantissa part, it is perfectly possible not to store this 1 before the decimal point, "1. " is also known as the hidden bit, so that the trailing part has an extra bit, thus improving the precision of the floating-point number, which in the IEEE-754 standard format is the data after the decimal point. For half-precision floating-point numbers, single-precision floating-point numbers and double-precision floating-point numbers are represented as shown in Table 2.1 [3-6].

In the IEEE-754 standard, for normalized floating-point numbers, its real value is represented as follows:

$$\mathrm{X} = (-1)^{S} \times M \times 2^{E-\alpha} \tag{2-1}$$

Where, S is sign bit, M is mantissa part, E is exponent part and $\alpha$ is offset value.

Table 2.1 Floating-point parameters

| Format | Half-precision | Single-precision | Double-precision |
|---|---|---|---|
| Sign | 1 | 1 | 1 |
| Exponential width | 5 | 8 | 11 |
| Mantissa width | 10 | 23 | 52 |

| | | | |
|---|---|---|---|
| Exponential offset | 15 | 127 | 1023 |
| Floating-point format width | 16 | 32 | 64 |

## 2.2 Basic algorithm operation

In this subsection, I briefly introduce the steps for using Newton-Raphson, CORDIC and Taylor-series expansion algorithms in arithmetic to give us some initial insight into these algorithms.

### 2.2.1 Newton-Raphson

The idea behind Newton Raphson's algorithm is to find the reciprocal of the divisor, then multiply it by the divisor, and finally find the final quotient [7, 8]. The multiplication operation can be calculated directly by calling the multiplier, so the difficulty of Newton-Raphson's algorithm becomes how to find the reciprocal of the divisor, as in the following equation:

$$\mathrm{y} = \frac{b}{c} = b \ \ \frac{1}{c} \tag{2-2}$$

For the reciprocal of a divisor, Newton Raphson's algorithm is found by multiple iterations. First given a function $f(x) = \frac{1}{x} - d,$ it is easy to see that when the function has function value of 0, the independent variable is the reciprocal of the divisor $d$. Starting from the image of the function, which is the intersection of the image of the function with the $x$-axis, is shown in Fig. 2.2.

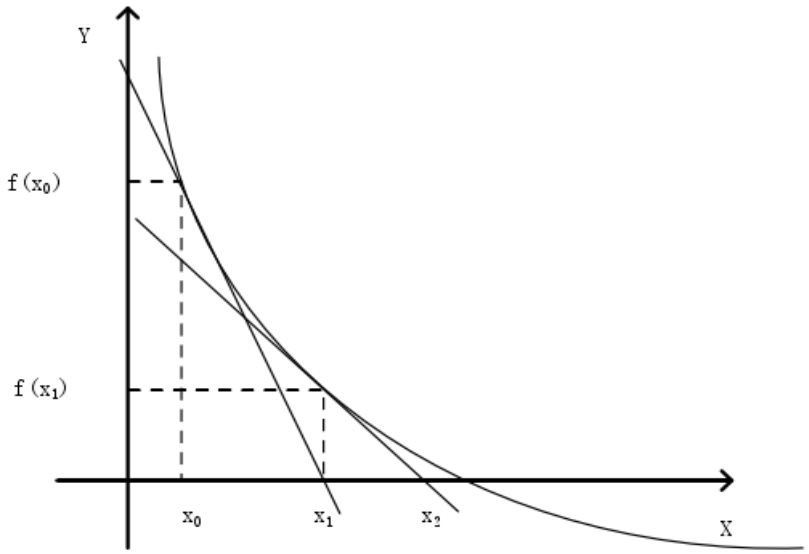



Fig. 2.2 Schematic diagram of the Newton Raphson algorithm iteration

As shown in Fig. 2.4, the objective is to determine the transverse coordinate X of the point where the image of the function intersects the $x$-axis. X is obtained by a number of $x$ iterations. The initial value of $x_0$ is determined, then the point on the image of the function with horizontal coordinate $x_0$ $(x_0,\ f(x_0))$ is found, and the tangent to the function is made at that point, and the intersection of the tangent and the $x$-axis is $x_1$. Then, with $x_1$ as the new initial value, the above iterative process is repeated to obtain $x_1,\ x_2,\ x_3,\ x_4\cdots$ in order to keep approximating *X*. By iterating several times, we can obtain the relationship between $x_0$ and $x_1$ as: $x_1 = x_0 - \frac{f(x_0)}{f(\acute{x}_0)}$. Then we continue to iterate, which can get the Newton Raphson algorithm iterative formula as:

$$x_i = x_{i-1} - \frac{f(x_{i-1})}{f(\acute{x}_{i-1})} \tag{2-3}$$

Because the derivative of the function $f(\acute{x}_{i-1}) = -\frac{1}{{x_{i-1}}^2}$, taking into

the above equation, we can obtain the following:

$$x_i = x_{i-1} - \frac{\frac{1}{x_i} - X}{-\frac{1}{{x_{i-1}}^2}} = x_{i-1}(2 - x_{i-1}X) \tag{2-4}$$

In numerical analysis, Newton Raphson's algorithm usually converges quickly, especially when the iterations start very close to the desired value. The closer the initial value is to the desired value the fewer iterations there are. The initial value is usually determined using a look-up table (LUT), based on which the initial value of the iteration is determined and then iterated.

### 2.2.2 CORDIC

The CORDIC algorithm is a well-known iterative technique for computing elementary functions that has been extensively studied and which uses only primitive arithmetic operations (addition, subtraction, shifting and lookup of stored functions), which is a great advantage in terms of hardware characteristics, especially when complex circuits are involved [9,10]. The basic idea of CORDIC is to perform two-dimensional vector processing using a series of micro-rotations (rotations at a predetermined angle). Therefore, The CORDIC algorithm belongs to the class of bit-by-bit algorithms with linear convergence, and the approximation requires approximately n iterations to obtain n bits of accuracy. A schematic of the CORIC algorithm rotation is shown in Fig. 2.3.

The CORDIC iterative equation is derived from the two-dimensional vector rotation mechanism. First, any rotation angle A is decomposed into a set of angles $\{A_i, i = 0, 1, 2, \cdots n\}$, that is $\theta = \sum_{i=0}^{n} \theta_i$ , where the

rotation direction parameter $\theta_i\{-1, 1\}$, and then the vector rotated by A is represented using the triangular equation as shown in following:

$$\begin{bmatrix} x_n \\ y_n \end{bmatrix} = \begin{bmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{bmatrix} \cdot \begin{bmatrix} x_0 \\ y_0 \end{bmatrix} = \prod_i^n cos\theta_i \begin{bmatrix} 1 & -\sigma_i tan\sigma_i \\ \sigma_i tan\sigma_i & 1 \end{bmatrix} \cdot \begin{bmatrix} x_0 \\ y_0 \end{bmatrix} \quad (2\text{-}5)$$

In order to use simple arithmetic operations to implement each matrix multiplication, the angle set is defined as the arctangent base, $\{\theta_i = tan^{-1}, i = 0, 1, 2, \cdots n\}$, thus the above formula is obtained.

$$\begin{bmatrix} x_n \\ y_n \end{bmatrix} = \prod_i^n cos\theta_i \begin{bmatrix} 1 & -\sigma_i tan\sigma_i \\ \sigma_i tan\sigma_i & 1 \end{bmatrix} \cdot \begin{bmatrix} x_0 \\ y_0 \end{bmatrix} \quad (2\text{-}6)$$

Based on the above equation, the CORDIC iterative equation for performing the vector rotation can be derived as follows:

$$\begin{cases} x_{i+1} = x_i - \sigma_i y_i 2^{-i} \\ y_{i+1} = y_i + \sigma_i x_i 2^{-i} \end{cases} \quad (2\text{-}7)$$

The CORDIC algorithm can be extended to evaluate various functions by using different modes of operation (rotational and vector) and coordinate systems (circular, linear and hyperbolic). In the rotation mode, the CORDIC processor performs a two-dimensional vector rotation. The processor produces a vector $(x_n, y_n)$ that converges to 0 with the angle $(z_n)$ output, obtained by rotating the angle input $(z_0)$. On the other hand, the vector mode processor calculates the magnitude $(x_n)$ and the angle of the initial vector $(x_0, y_0)$ converges to 0 with $y_n$. Considering the two modes of operation with the three coordinate systems acting together, the unified CORDIC iterative equation for step *i* is expressed as the following equation:

$$\begin{cases} x_{i+1} = x_i - m\sigma_i y_i 2^{-i} \\ y_{i+1} = y_i + \sigma_i x_i 2^{-i} \\ z_{i+1} = z_i - \sigma_i \alpha_i \end{cases} \tag{2-8}$$

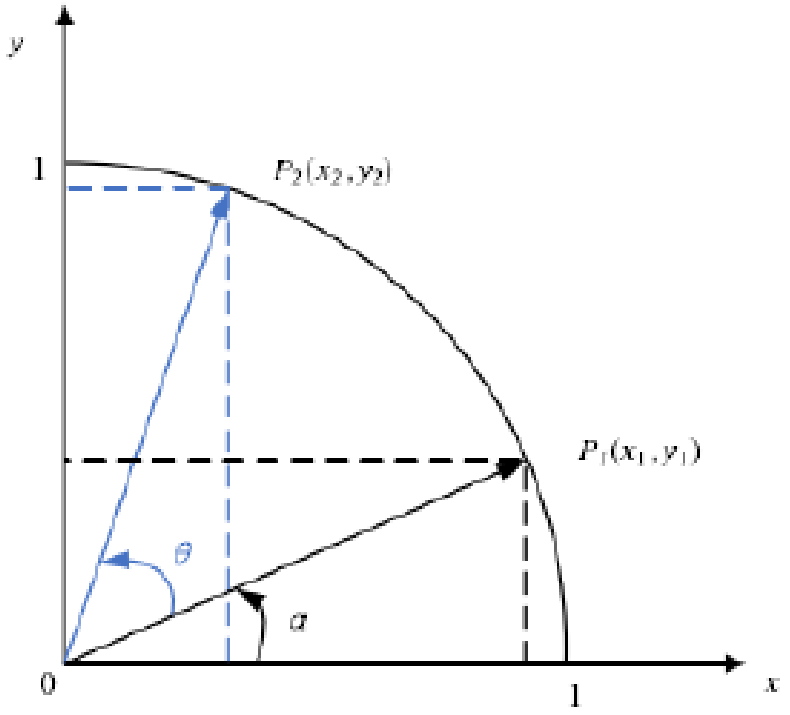



Fig. 2.3 schematic of the CORIC algorithm rotation

Here $m$ is equal to 1, 0 and -1 in the circular, linear and hyperbolic coordinate systems, respectively, and $\sigma_i$ is defined as $sign(z_i)$, which is defined in the rotational or vector model as $\text{sign}(x_i, y_i)$ and the angle of rotation is $\alpha_i = m^{-\frac{1}{2}} tan^{-1}(m^{\frac{1}{2}} 2^{-i})$. Iterations start from $i$ =0 when $m$=1 and from $i \geq 1$ when m=0 or -1. In order to satisfy CORDIC convergence, we need to satisfy the following equation:

$$\forall \text{i}, \propto_i \leq \sum_{j=i+1}^{\infty} \propto_j \tag{2-9}$$

When m = -1, the iteration needs to be repeated twice at $i = 4, 13, \cdots, j, 3j + 1$ in order to satisfy the condition upper equation. At the same time, the norm of the vector is amplified by the scale factor $K_m = \prod_{i=0}^{n-1} \sqrt{1 + m \cdot 2^{-2i}}$ during the CORDIC rotation. Proportionality factor compensation is usually accomplished by post-multiplication and this proportionality factor can be effectively removed. The final result is that when the appropriate initial vectors $(x_0, y_0)$ are input and the correct mode of operation and coordinate system are chosen, the CORDIC algorithm can be used to evaluate trigonometric and transcendental primitive functions.

### 2.2.3 Taylor-series expansion

The Taylor-series expansion algorithm is similar to the aforementioned Newton Raphson algorithm in that it converts the division of the divisor and the divisor into the multiplication of the reciprocal of the divisor and the reciprocal of the divisor [11,12]. The difference is that in the process of finding the reciprocal of the divisor, the Taylor-series expansion algorithm uses a Taylor-series expansion to approximate the reciprocal value. The following section provides a brief introduction to Taylor-series expansions.

In mathematics, a Taylor-series is a representation of a function that is the sum of an infinite number of terms calculated from the value of the derivative at a point. Let $f(x)$ be a differentiable function corresponding to an algebraic expression, then the Taylor-series of $f(x)$ is defined as:

$$f(x) = \sum_{n=0}^{\infty} \frac{1}{n!} (x - x_0)^n f^n(x_0) \tag{2-10}$$

Another format we can express is as follows:

$$f(x) = f(x_0) + xf'(x_0) + \frac{x^2}{2!}(f)''(x_0) + \cdots + \frac{(x - x_0)}{n!}(f)^n(x_0) + R_n(x) \tag{2-11}$$

where $f'(x_0)$, $f''(x_0)$, etc. are the first, second and higher order derivatives of $f(x)$ and $R_n(x)$ is the Lagrangian remainder. The error R is bounded as follows:

$$R_n(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!}(x - x_0)^{n+1} \tag{2-12}$$

If the value of a function and all its derivatives at a point are known, the Taylor-series can be used to calculate the value of the entire function at each point. The partial sum (Taylor polynomial) of the series can be used as an approximation for the whole function.

## 2.3 Summary

This chapter mainly introduces the floating-point number representation method and format, and introduces the more commonly used algorithms, including Newton's iteration method, CORDIC algorithm and Taylor-series expansion method. Various algorithms can complete arithmetic operations. Where should we consider choosing the best algorithm? This needs to take into account the hardware area occupied by the algorithm, the time delay caused by the algorithm, the speed of operation and the accuracy of the algorithm. For researchers to work on this series of issues, the hardware resource occupancy rate is lower and the speed is faster; this design is also developed around these issues.

# Reference


[1] V. Rajaraman, "IEEE Standard for Floating-point Numbers", Resonance, vol. 21, no.1, pp. 11-30 (Jun. 2016).

[2] P. Markstein, "The New IEEE-754 Standard for Floating-point Arithmetic", Dagstuhl Seminar Proceedings, Schloss Dagstuhl-Leibniz-Zentrum für Informatik, Germany (Apr. 2008).

[3] C. R. Aguilera-Galicia, O. Longoria-Gandara, O. A. Guzmán-Ramos, L. Pizano-Escalante and J. Vázouez-Castillo, "IEEE-754 Half-Precision Floating-Point Low-Latency Reciprocal Square Root IP-Core", IEEE 10th Latin-American Conference on Communications, Guadalajara, Mexico (Nov. 2018).

[4] M. Shirke, S. Chandrababu and Y. Abhyankar, "Implementation of IEEE 754 Compliant Single Precision Floating-point Adder Unit Supporting Denormal Inputs on Xilinx FPGA", IEEE International Conference on Power, Control, Signals and Instrumentation Engineering, Chennai, India (Sept. 2017).

[5] K. Manolopoulos, D. Reisis and V. A. Chouliaras, "An Efficient Multiple Precision Floating-Point Multiplier", 18th IEEE International Conference on Electronics, Circuits, and Systems, Beirut, Lebanon (Dec. 2011).

[6] A. Rahman, Abdullah-Al-Kafi, M. Khalid, A. T. M. S. Islam and M. Rahman, "Optimized Hardware Architecture for Implementing IEEE 754 Standard Double Precision Floating-Point Adder/Subtractor", 17th International Conference on Computer and Information Technology, Dhaka, Bangladesh (Dec. 2014).

[7] L-K Wang and M. J. Schulte, "Decimal Floating-point Division Using Newton-Raphson Iteration", 15th IEEE International Conference on Application-Specific Systems, Architectures and Processors, Galveston, TX (Sept. 2004).

[8] N. Louvet, J. Muller and A. Panhaleux, "Newton-Raphson Algorithms for Floating-point Division Using An FMA", 21st IEEE International Conference on Application-specific Systems, Architectures and Processors, Rennes, France (Jul. 2010).

[9] G. J. Hekstra and E. F. A. Deprettere, "Floating-point CORDIC", IEEE 11th Symposium on Computer Arithmetic, Windsor, ON, Canada (Jul. 1993).

[10] H.-T. Nguyen, X.-T. Nguyen, T.-T. Hoang, D.-H. Le and C.-K. Pham, "Low-Resource Low-Latency Hybrid Adaptive CORDIC With Floating-Point Precision", IEICE Electron. Exp., vol. 12, no. 9, (Apr. 2015).

[11] T-J Kwon, J. Sondeen and J. Draper, "Floating-Point Division and Square Root Using A Taylor-series Expansion Algorithm", 50th Midwest Symposium on Circuits

and Systems, Montreal, QC, Canada (Aug. 2007).

[12] Taek-Jun Kwon and J. Draper, "Floating-point Division and Square Root Implementation Using a Taylor-series Expansion Algorithm With Reduced Look-Up Tables", 51st IEEE Midwest Symposium on Circuits and Systems, Knoxville, TN, (Aug. 2008).

# Chapter 3

# REVISIT TO FLOATING-POINT DIVISION ALGORITHM BASED ON TAYLOR-SERIES EXPANSION

## 3.1 Abstract

This paper investigates floating-point division algorithms based on Taylor-series expansion. Taylor-series expansions of $1/x$ are examined for several center points with their convergence ranges, and we show the Taylor-series expansion division algorithm trade-offs among division accuracy, numbers of multiplications/additions/subtractions and LUT sizes; the designer can choose the optimal algorithm for his digital division, and build its conceptual architecture design with the contents described here.

Keywords—Floating-point Division Algorithm, Digital Divider, Taylor-series Expansion, Digital Arithmetic

## 3.2 Introduction

Due to constant advances in VLSI technology, binary floating-point representation becomes more important and high precision and high-speed floating-point calculation in embedded systems and mobile applications becomes possible. There calculations of addition, subtraction, multiplication and division are involved: addition and subtraction are relatively easy to implement, and multiplication is complicated to implement, whereas division is very complex to implement.

In this paper, we focus a simple-yet-accurate floating-point division algorithm using Taylor-series expansion of $f(x) = 1/x$. There are many division algorithms including: table lookup, functional iteration, variable latency, high radix, and digit recurrence. More detailed concepts about division can be found in [1]. The floating-point division calculation is of very importance role in the range of signal processing of hardware implementation. It is mentioned in [1, 2, 3] that the hardware complexity can be controlled and the calculation delay (latency) is small, but the details such as the accuracy of Taylor-series expansion seem not to be disclosed. Here we use the convergence of Taylor-series expansion of $f(x) = 1/x$ to show trade-offs among division accuracy, numbers of multiplications/additions/subtractions and LUT sizes so that the designer can choose the optimal algorithm for his digital division.

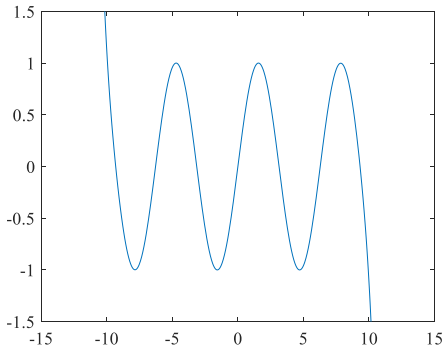
Compared with the Newton-Raphson method and digit recurrence method, our investigating method can improve the calculation efficiency and low latency of division in LUT (look up table), and can also control the numbers of multiplications/additions/subtractions and the LUT size, according to the requirements of the designer. We show some numerical simulation results and its hardware implementation considerations.

### 3.3 Taylor-series Expansion

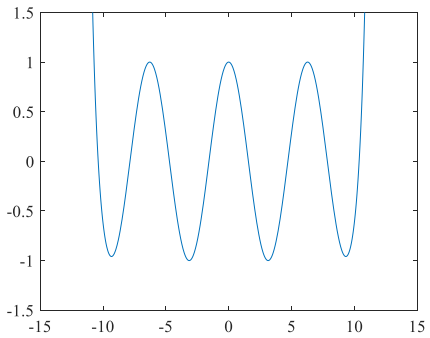
Consider an infinitely differentiable function f(x). Its Taylor-series expansion at $x = a$ is given by:

$$f(x) = \mathrm{f(a)} + f'(a)(x-a) + \frac{(\mathrm{f})''(a)}{2!}(x-a)^2 + \cdots + \frac{(\mathrm{f})^n(a)}{\mathrm{n!}}(x-a)^{\mathrm{n}} + \cdots \tag{3-1}$$

Notice that many functions used in engineering design have relatively wide convergence radius; as *n* increases, the Taylor-series expansion reaches $f(x)$ for wide range of *x*. For examples, Taylor-series expansions (with $a$ = 0) of sine and cosine functions converge to sine and cosine for $-\infty < x < \infty$, respectively, as *n* increases (Fig. 3.1).




(a) sin(*x*)




(b) cos(*x*)

Fig. 3.1 Waveforms of Taylor-series expansion of sin(*x*) and cos(*x*) at *a*=0 up-to 25 terms.

However, as far as we know, there are very few algorithms in the engineering field that positively utilize the fact that Taylor-series expansion of some $f(x)$ converges to $f(x)$ over a wide range of $x$. Then we investigate here a digital arithmetic algorithm for the reciprocal calculation of $x$ by approximating $f(x) = 1/x$ by Taylor-series expansion.

## 3.4 Investigated Division Algorithm

### 3.4.1 Problem Formulation

Consider floating-point representation in binary.

Mantissa : M

Exponent : E

Binary representation : $\mathrm{M} \times 2^{\mathrm{E}}$

Mantissa $\mathrm{M} = 1.\alpha\beta\gamma\cdots$ (Here $\alpha, \beta, \gamma\ldots$ is 0 or 1)

Notice that the binary point is put so that $1 \leq \mathrm{M} < 2$. For example, M=1.011001(binary) = 1.390625 (decimal).

Now let us consider a division algorithm that calculates A=N/D for two numbers N and D in binary floating-point representation, where $A$, N and $D$ are expressed as follows:

$$\begin{aligned} \mathrm{A} &= M_A \times 2^{E_A} \\ N &= M_N \times 2^{E_N} \\ D &= M_D \times 2^{E_D} \end{aligned} \tag{3-2}$$

If 1/D is calculated, A=N/D can be calculated using a conventional digital multiplication algorithm. Since 1/D is equal to $1/\mathrm{M_D} \times 2^{-E}$, the

mantissa reciprocal calculation ($1/M_D$) is the important point to obtain 1/D. Then we study an algorithm that performs Taylor-series expansion of $f(x) = 1/x$ and its approximated calculation with the desired accuracy.

### 3.4.2 Reciprocal calculation by Taylor-series expansion

Let us consider the calculation of $1/M_D$ ($1 \le M_D < 2$) using Taylor-series expansion of $f(x) = 1/x$ with $x$=$a$ ($1 \le a < 2$) satisfying the desired accuracy. Taylor-series expansion of $f(x) = 1/x$ with $x = a$ is as follows:

$$f(x) = \frac{1}{a} - (x-a) + (x-a)^2 - (x-a)^3 + (x-a)^4 + \cdots + (-1)^n (x-a)^n + \cdots \quad (3\text{-}3)$$

Notice that the coefficient of each term is +1 or -1, which simplifies the calculation by reducing the number of multiplications.

Fig. 3.2 shows the convergence range of Taylor-series expansion of $f(x) = 1/x$ with different $a > 0$. We can conjecture that the convergence range for a > 0 is 0 < x < 2a.

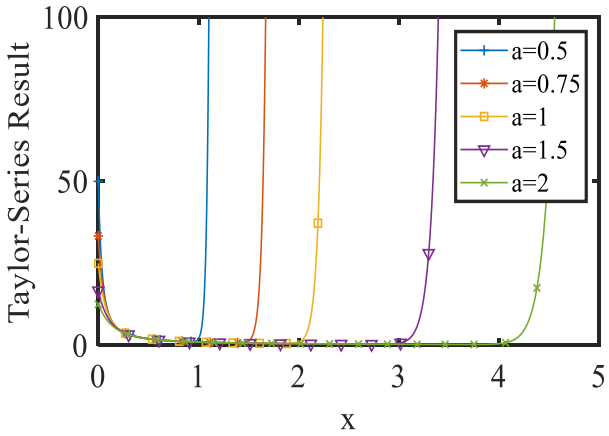



Fig. 3.2 Convergence range of Taylor-series expansion with different $a$ at $f(x) = 1/x$ (number of terms: 25).

### 3.5 Simulation Results

In this section, we use simulation to obtain the number of terms *n* expanded by Taylor-series of $f(x) = 1/x$ in different cases of given accuracies and given regions, which satisfies the following:

| {$f(x)$ – [Taylor-series expansion of $f(x)$ with n terms]}/ $f(x)$ | < given accuracy for all x in given region.

#### 3.5.1 Binary point position of mantissa $M_D$=1.$\alpha\beta\gamma\cdots$ (1≤ $M_D$<2)

(1) One Region of 1≤ $M_D$<2

Table1 shows the required number of the terms *n* for Taylor-series expansion to meet the desired accuracy.

A-1-a) Taylor-series expansion of $f(x) = 1/x$ at *a* = 1.5 (1≤*x*<2)

For example, when the given accuracy is $1/2^{16}$, we obtain *n*=11 from numerical simulation which satisfied the following:

| {$f(x)$ – [Taylor-series expansion of $f(x)$ with n terms]}/ $f(x)$ | < $1/2^{16}$ for all *x* (1≤ *x* <2)

Table 3.1 shows the numerical calculation results.

Table 3.1. Number of Taylor-series expansion terms that meets specified accuracy for one region of 1≤ *x* <2.

| Precision / Taylor-series expansion | $\frac{1}{2^8}$ | $\frac{1}{2^{16}}$ | $\frac{1}{2^{20}}$ | $\frac{1}{2^{24}}$ | $\frac{1}{2^{32}}$ |
|---|---|---|---|---|---|
| A-1-a) | 6 | 11 | 13 | 16 | 21 |

(2) Division by 2 of region 1≤ *x* <2.

Divide the region $1 \leq x < 2$ by 2 and choose the region, based on $M_D$. Then perform the Taylor-series expansion at the point $a$ of the center of each divided region.

A-2-a) For $M_D$ = 1.0xxxxx⋯ $(1 \leq M_D < 1.5)$, $a$=1.25.

A-2-b) For $M_D$ = 1.1xxxxx⋯ $(1.5 \leqq M_D < 2)$, $a$=1.75.

Table 3.2 shows the numerical simulation results.

Table 3.2 Number of Taylor-series expansion terms that meets specified accuracy when the region of $1 \leq x < 2$ is divided by 2.

| Precision / Taylor-series expansion | $\frac{1}{2^8}$ | $\frac{1}{2^{16}}$ | $\frac{1}{2^{20}}$ | $\frac{1}{2^{24}}$ | $\frac{1}{2^{32}}$ |
|---|---|---|---|---|---|
| A-2-a) | 4 | 7 | 9 | 11 | 14 |
| A-2-b) | 3 | 6 | 8 | 9 | 12 |

(3) Division by 4 of region $1 \leq x < 2$.

Divide the region $1 \leq x < 2$ by 4 and choose the region, based on $M_D$. Then perform the Taylor-series expansion at the point $a$ of the center of each divided region.

A-3-a) For $M_D$ = 1.00xxxxx⋯ $(1 \leq M_D < 1.25)$, $a$ =1.125.

A-3-b) For $M_D$ = 1.01xxxxx⋯ $(1.25 \leq M_D < 1.5)$, $a$ =1.375.

A-3-c) For $M_D$ = 1.10xxxxx⋯ $(1.5 \leq M_D < 1.75)$, $a$ =1.625.

A-3-d) For $M_D$ = 1.11xxxxx⋯ $(1.75 \leq M_D < 2)$, $a$ =1.875.

Table 3.3 shows the numerical simulation results.

Table 3.3 Number of Taylor-series expansion terms that meets specified accuracy when the region of $1 \leq x < 2$ is divided by 4.

| Precision / Taylor-series expansion | $\frac{1}{2^8}$ | $\frac{1}{2^{16}}$ | $\frac{1}{2^{20}}$ | $\frac{1}{2^{24}}$ | $\frac{1}{2^{32}}$ |
|---|---|---|---|---|---|
| A-3-a) | 3 | 6 | 7 | 8 | 11 |
| A-3-b) | 3 | 5 | 6 | 7 | 10 |
| A-3-c) | 3 | 5 | 6 | 7 | 9 |
| A-3-d) | 3 | 5 | 6 | 7 | 9 |

(4) Division by 8 of region $1 \leq x < 2$.

Divide the region $1 \leq x < 2$ by 8 and choose the region, based on $M_D$. Then perform the Taylor-series expansion at the point $a$ of the center of each divided region.

A-4-a) For $M_D$ = 1.000xxxx… $(1 \leqq M_D < 1.125)$, $a$ =1.0625.

A-4-b) For $M_D$ = 1.001xxxx… $(1.125 \leq M_D < 1.25)$, $a$ =1.1875.

A-4-c) For $M_D$ = 1.010xxxx… $(1.25 \leqq M_D < 1.375)$, $a$ =1.3125.

A-4-d) For $M_D$ = 1.011xxxx… $(1.375 \leq M_D < 1.5)$, $a$ =1.4375.

A-4-e) For $M_D$ = 1.100xxxx… $(1.5 \leq M_D < 1.625)$, $a$ =1.5625.

A-4-f) For $M_D$ = 1.101xxxx… $(1.625 \leq M_D < 1.75)$, $a$ =1.6875.

A-4-g) For $M_D$ = 1.110xxxx… $(1.75 \leq M_D < 1.875)$, $a$ =1.8125.

A-4-h) For $M_D$ = 1.111xxxx… $(1.875 \leq M_D < 2)$, $a$ =1.9375.

Table 3.4 shows the numerical simulation results.

Table 3.4 Number of Taylor-series expansion terms that meets specified accuracy when the region of $1 \leq x < 2$ is divided by 8.

| Precision / Taylor-series expansion | $\frac{1}{2^8}$ | $\frac{1}{2^{16}}$ | $\frac{1}{2^{20}}$ | $\frac{1}{2^{24}}$ | $\frac{1}{2^{32}}$ |
|---|---|---|---|---|---|
| A-4-a) | 2 | 4 | 5 | 6 | 8 |
| A-4-b) | 2 | 4 | 5 | 6 | 8 |
| A-4-c) | 2 | 4 | 5 | 6 | 8 |
| A-4-d) | 2 | 4 | 5 | 6 | 8 |
| A-4-e) | 2 | 4 | 5 | 6 | 7 |
| A-4-f) | 2 | 4 | 5 | 6 | 7 |
| A-4-g) | 2 | 4 | 5 | 5 | 7 |
| A-4-h) | 2 | 4 | 5 | 5 | 7 |

### 3.5.2 Binary point position of mantissa $M_D = 0.1\alpha\beta\gamma\ldots$ $(1/2 \leq M_D < 1)$

(1) One Region of $1/2 \leq M_D < 1$

Let us consider the case that the mantissa part $M_D$ of the denominator D is expressed as $0.1\alpha\beta\gamma\cdots$ $(1/2 \leq M_D < 1)$. Table 3.5 shows the numerical simulation result of the required number n of Taylor-series expansion terms.

B-1-a) For $M_D = 0.1xxxx\cdots$ $(0.5 \leqq M_D < 1)$, a=0.75.

Table 3.5 Number of Taylor-series expansion terms that meets specified accuracy for one region of $1/2 \leq x < 1$.

| Precision | $\frac{1}{2^8}$ | $\frac{1}{2^{16}}$ | $\frac{1}{2^{20}}$ | $\frac{1}{2^{24}}$ | $\frac{1}{2^{32}}$ |
|---|---|---|---|---|---|

| Taylor-series expansion | | | | | |
|---|---|---|---|---|---|
| B-1-a) | 6 | 11 | 13 | 16 | 21 |

(2) Division by 2 of region $1/2 \leq x < 1$.

B-2-a) For $M_D = 0.10xxxxx\cdots$ $(0.5 \leq M_D < 0.75)$, $a = 0.625$.

B-2-b) For $M_D = 0.11xxxxx\cdots$ $0.7.5 \leqq M_D < 1)$, $a = 0.875$.

Table 3.6 shows the numerical simulation results.

Table 3.6 Number of Taylor-series expansion terms that meets specified accuracy when the region of $1/2 \leq x < 1$ is divided by 2.

| Precision / Taylor-series expansion | $\frac{1}{2^8}$ | $\frac{1}{2^{16}}$ | $\frac{1}{2^{20}}$ | $\frac{1}{2^{24}}$ | $\frac{1}{2^{32}}$ |
|---|---|---|---|---|---|
| B-2-a) | 4 | 7 | 9 | 11 | 14 |
| B-2-b) | 3 | 6 | 8 | 9 | 12 |

(3) Division by 4 of region $1/2 \leq x < 1$.

B-3-a) For $M_D = 0.100xxxxx\cdots$ $(0.5 \leq M_D < 0.625)$, $a = 0.5625$.

B-3-b) For $M_D = 0.101xxxxx\cdots$ $(0.625 \leq M_D < 0.75)$, $a = 0.7125$.

B-3-c) For $M_D = 0.110xxxxx\cdots$ $(0.75 \leq M_D < 0.875)$, $a = 0.8125$.

B-3-d) For $M_D = 0.111xxxxx\cdots$ $(0.875 \leq M_D < 1)$, $a = 0.9375$.

Table 3.7 shows the numerical simulation results.

Table 3.7 Number of Taylor-series expansion terms that meets specified accuracy when the region of $1/2 \leq x < 1$ is divided by 4.

| Taylor-series expansion \ Precision | $\frac{1}{2^8}$ | $\frac{1}{2^{16}}$ | $\frac{1}{2^{20}}$ | $\frac{1}{2^{24}}$ | $\frac{1}{2^{32}}$ |
|---|---|---|---|---|---|
| B-3-a) | 3 | 6 | 7 | 8 | 11 |
| B-3-b) | 3 | 5 | 6 | 7 | 10 |
| B-3-c) | 3 | 5 | 6 | 7 | 9 |
| B-3-d) | 3 | 5 | 6 | 7 | 9 |

(4) Division by 8 of region $1/2 \leq x < 1$.

B-4-a) For $M_D$ = 0.1000xxxx⋯ ($0.5 \leqq M_D < 0.5625$), $a$=0.5313.

B-4-b) For $M_D$ = 0.1001xxxx⋯ ($0.5625 \leq M_D < 0.625$), $a$=0.5938.

B-4-c) For $M_D$ = 0.1010xxxx⋯ ($0.625 \leqq M_D < 0.6875$), $a$=0.6563.

B-4-d) For $M_D$ = 0.1011xxxx⋯ ($0.6875 \leq M_D < 0.75$), $a$=0.7188.

B-4-e) For $M_D$ = 0.1100xxxx⋯ ($0.75 \leq M_D < 0.8125$), $a$=0.7813.

B-4-f) For $M_D$ = 0.1101xxxx⋯ ($0.8125 \leq M_D < 0.875$), $a$=0.8438.

B-4-g) For $M_D$ = 0.1110xxxx⋯ ($08.75 \leq M_D < 0.9375$), $a$=0.9063.

B-4-h) For $M_D$ = 0.1111xxxx⋯ ($0.9375 \leq M_D < 1$), $a$=0.9688.

Table 3.8 shows the numerical simulation results.

Table 3.8 Number of Taylor-series expansion terms that meets specified accuracy when the region of $1/2 \leq x < 1$ is divided by 8.

| Precision / Taylor-series expansion | $\frac{1}{2^8}$ | $\frac{1}{2^{16}}$ | $\frac{1}{2^{20}}$ | $\frac{1}{2^{24}}$ | $\frac{1}{2^{32}}$ |
|---|---|---|---|---|---|
| B-4-a) | 2 | 4 | 5 | 6 | 8 |
| B-4-b) | 2 | 4 | 5 | 6 | 8 |
| B-4-c) | 2 | 4 | 5 | 6 | 8 |
| B-4-d) | 2 | 4 | 5 | 6 | 8 |
| B-4-e) | 2 | 4 | 5 | 6 | 7 |
| B-4-f) | 2 | 4 | 5 | 6 | 7 |
| B-4-g) | 2 | 4 | 5 | 5 | 7 |
| B-4-h) | 2 | 4 | 5 | 5 | 7 |

We see from Tables 3.4 and 3.8 that the number of Taylor-series expansion terms required to obtain the desired precision is the same whether the mantissa $M_D$ is $1.\alpha\beta\gamma\cdots$ ($1 \leq M_D < 2$) or $0.1\alpha\beta\gamma\cdots$ ($1/2 \leq M_D < 1$). For example, when the Taylor-series expansion precision is 24-bit, in both $M_D$cases, the required number $n$ of terms is 6 for the region division by 8, and it is 5 for the division by 16 while it is 4 for the division of 32.

## 3.6 Hardware Implementation Consideration

Let us consider the hardware implementation complexity for performing the reciprocal calculation of $f(x) = 1/x$ with the investigated algorithm. Notice that each term coefficient of the Taylor-series expansion of $f(x) = 1/x$ is +1 or -1 which appears alternately, and then we can reduce the number of multiplications for the Taylor-series calculation.

### 3.6.1 Required numbers of multiplications/additions/subtractions for Taylor-series expansion

***Example 1:*** Five terms of Taylor-series expansion case

Let us consider to calculate the following with multiplier/ additions/subtracter and LUT.

$$f_5 = (1/a) - (x-a) + (x-a)^2 - (x-a)^3 + (x-a)^4 \qquad (3\text{-}4)$$

Where *a* is a constant and *x* is a variable. 1/*a* is calculated in advance and stored in LUT memory, which is read for the calculation. Then we calculate $y = x - a$ and $z = y^2$. Next, we calculate the followings:

$$\begin{aligned} f_5 &= \left(\frac{1}{a}\right) - y + y^2 - y^3 + y^4 \\ &= (1/a) - (y-z)(1+z) \end{aligned} \qquad (3\text{-}5)$$

We see that $f_5$ can be obtained with 2 multiplications and 4 additions/subtractions.

We see from Tables 3.4 and 3.8 that by dividing the region by 8, the reciprocal of the mantissa can be calculated with 20-bit accuracy by 2 multiplications and 4 additions/subtractions.

***Example 2:*** Seven terms of Taylor-series expansion case

Next, let us consider the following:

$$\begin{aligned} f_7 = (1/a) - (x-a) + (x-a)^2 - (x-a)^3 + (x-a)^4 \\ - (x-a)^5 + (x-a)^6 \end{aligned} \qquad (3\text{-}6)$$

Similarly, (1/a) is read from LUT and we calculate $y=x-a$ and $z=y^2$.

Then we calculate the following:

$$f_7 = (1/a) - y + y^2 - y^3 + y^4 - y^5 + y^6 \\ = (1/a) - (y - z)(1 + z + z^2) \quad (3\text{-}7)$$

We see that $f_7$ can be obtained with 3 multiplications and 5 additions/subtractions.

Table 3.9 shows the required numbers of multiplications and additions/subtractions for the number of Taylor-series terms for $f(x) = 1/x$.

### 3.6.2 LUT contents and size

(1/a) is calculated in advance stored in memory. It is then read at Taylor-series calculation time. In case that the region is divided by 4, 4-word LUT is required as shown in Table 3.10, and the data at the address αβ is read for *M=1*.αβ⋯.

### 3.6.3 Comparison of original and investigated methods

The original Taylor-series expansion division method is described in [1, 2, 3], but its design details seem not to be disclosed. Our investigated method directly calculates the reciprocal of the divider mantissa $M_D$ using Taylor-series expansion, which may be one of different points. At least, we estimate that the number of multiplications is not larger and the LUT size is not bigger for our investigated method; the detailed comparison is left for the future work. Also, we would like to just claim that the designer can build his/her conceptual divider architecture design with the contents described in this paper.

## 3.7 Summary

We have studied floating-point division algorithms with Taylor-series expansion of $f(x) = 1/x$, and show their hardware implementation trade-offs among division accuracy, numbers of multiplications/additions /subtractions and LUT sizes to meet various digital division specifications flexibly. The designer can implement his/her floating-point division with these contents described in this paper.

Table 3.9 Required numbers of multiplications and additions/subtractions for *n*-term Taylor-series expansion.

| # of Taylor-series expansion terms | # of multiplications | # of additions and subtractions |
|---|---|---|
| 3 | 1 | 3 |
| 4 | 2 | 4 |
| 5 | 2 | 4 |
| 6 | 3 | 5 |
| 7 | 3 | 5 |
| 8 | 4 | 6 |

Table 3.10 LUT memory for region division by 4.

| Address (αβ) | LUT data |
|---|---|
| 00 | Reciprocal of $a$ = 1.125 |
| 01 | Reciprocal of $a$ = 1.357 |

| | |
|---|---|
| 10 | Reciprocal of $a = 1.625$ |
| 11 | Reciprocal of $a = 1.875$ |

# References

[1] S. E. Obermann and M. J. Flynn. “Division Algorithms and Implementations”, IEEE Transactions on Computers, vol. 46, no. 8, pp. 833-854 (Aug.1997).

[2] T.-J. Kwon, J. Sondeen, J. Draper, “Floating-point Division and Square Root Using A Taylor-series Expansion Algorithm”, 50th Midwest Symposium on Circuits and Systems, Montreal, QC, Canada (Aug. 2007).

[3] A. Liddicoat, M. J. Flynn, “High Performance Floating Divide”, Euromicro Symposium on Digital Systems Design, Warsaw, Poland (Jan. 2001).

# Chapter 4

# FLOATING-POINT SQUARE ROOT CALCULATION ALGORITHM BASED ON TAYLOR-SERIES EXPANSION AND REGION DIVISION

## 4.1 Abstract

This paper describes details of floating-point square root calculation algorithms using Taylor-series expansion and mantissa region division for the efficient dedicated hardware design. Taylor-series expansions of the square root are examined at the center points of divided mantissa regions, and based on these, square root calculation hardware design balances among accuracy, numbers of basic floating-point arithmetic operations (multiplications and additions/ subtractions) as well as the required look-up table size are clarified quantitatively. These results lead to obtaining the suitable algorithm for floating-point square root calculation, and building its corresponding hardware architecture for the digital processor designer.

Keywords — Floating-point, Square Root, Taylor-series Expansion, Digital Arithmetic, Region Division

## 4.2 Introduction

The rapid advancement of Internet-of-Things (IoTs) service confirms

that it is one of the most important avenues to future technologies and it gains vast attention from a wide range of industries [1]. Floating-point arithmetic plays a crucial role in IoT service, and it prevails in lots of applications such as high precision signal processing and scientific computing calculations [2]-[4]. In addition to fundamental arithmetic operations such as addition, subtraction and multiplication as well as division, the square root determination (SRD) is often required for realizing complicated signal processing and scientific computing algorithms. Unfortunately, SRD creates many constraints in terms of hardware resources and power efficiency [5].

We have discussed division and inverse square root algorithms using Taylor-series expansion (TSE) [6], [7]. This chapter focuses on square root calculation algorithms in binary floating-point format using TSE with mantissa region division; these are simple yet accurate algorithms due to the mantissa region division method. There are plenty of algorithms for SRD such as table lookup, functional iteration, variable latency and high radix as well as digit recurrence methods in [8]-[10]. It is described in [11], [12] that complexity of arithmetic algorithm calculation hardware can be flexible with the minimized calculation latency, but the design details such as the TSE calculation errors remain to be determined. For high-speed and low-power, a shared partition SRD architecture is proposed in [13], where GST (generalized Svoboda and Tung) and SRT (Sweeney, Robertson and Tocher) are used to achieve high speed and low power. Table-look algorithms are introduced in [14] for calculating elementary functions; their analytical studies have shown that they provide higher speed and accuracy, compared

to the original methods. Many other papers have presented results in this area [15]-[17].

In this chapter we base SRD on the convergence characteristics of TSE with mantissa region division and clarify the design balance among arithmetic accuracy, numbers of basic arithmetic operations (multiply, add, subtract) and look-up table (LUT) size. Our work here helps the digital processor designer to select the suitable algorithm for his/her digital SRD design target flexibly and realize it as dedicated hardware as well as FPGA. Our proposed method can offer improved calculation efficiency, and also flexible control over amount of the required hardware. We present implementation hardware complexity considerations supported by some simulation results.

### 4.3 Taylor-series Expansion

Let us consider a function $f(x)$, which is infinitely differentiable. Then its Taylor-series expansion (TSE) at the center of $a$ is expressed by:

$$f(x) = f(a) + f'(a)(x-a) + \frac{(f)''(a)}{2!}(x-a)^2 + \cdots + \frac{(f)^n(a)}{n!}(x-a)^n \tag{4-1}$$

In general, as the number of terms *n* increases, the TSE approaches the exact value $f(x)$ for *x* in the convergence radius.

## 4.4 Square Root Calculation Algorithm

### 4.4.1 Problem Formulation

In general, the mantissa and the exponent as well as the sign bit can be used to form a floating-point number. Here, a positive number $X$ in the binary floating-point format is considered, and hence the sign-bit is always "positive", which is ignored. Let $X$ be the floating-point representation, while let $M$ and $E$ be its mantissa part, and the exponent part in binary representation, respectively. Then we have the following relationships:

$$X = M \times 2^E \tag{4-2}$$

Here, the mantissa is expressed by $M = 1.ABCD\cdots$ $(A, B, C, D, \cdots$ is 0 or 1). For the binary floating-point number, the range of the mantissa $M$ is in [1, 2); in other words $1 \leq M < 2$, such as, $M$=1.101101 (binary) , which is equal to 1.703125 (decimal).

For the square root calculation with the binary floating-point number, we can use the following expression:

$$S = \sqrt{X} = \sqrt{M} \times \sqrt{2^E} \tag{4-3}$$

### 4.4.2 Square Root Calculation of Exponent Part

Here, the exponent part calculation is discussed in two cases that the exponent part $E$ is even or odd.

1) In case that $E$ is even: Let $E = 2k$, and we have the following:

$$\sqrt{2^E} = 2^k \tag{4-4}$$

$$S = \sqrt{M} \times 2^k \tag{4-5}$$

We have the mantissa part $\sqrt{M}$ and the exponent part $k$ of $S$. Then normalize the floating-point expression of $\sqrt{M}$, and obtain the following:

$$\sqrt{M} = \ M_1 \times 2^{k_1} \tag{4-6}$$

Since $1 \leq M < 2$, then $1 \leq M_1 < \sqrt{2}$ and hence $k_1 = 0$. We have the following:

$$S = M_1 \times 2^k \tag{4-7}$$

We have the mantissa part $M_1$ and the exponent part $k$ of the final expansion $S$.

2) In case that $E$ is odd: Let $E = 2k + 1$, and we have the following:

$$\sqrt{2^E} = \sqrt{2} \times 2^k \tag{4-8}$$

Normalize the floating-point expression of $\sqrt{2} \times \sqrt{M}$, and obtain the following expression:

$$\sqrt{2} \times \sqrt{M} = \ M_2 \times 2^{k_2} \tag{4-9}$$

Since $1 \leq M < 2$, then $\sqrt{2} \leq \sqrt{2} \times \sqrt{M} < 2$, which leads to $\sqrt{2} \leq$

$M_2 < 2$ and $k_2 = 0$, and we have the following:

$$S = M_2 \times 2^k \tag{4-10}$$

We have the mantissa part $M_2$ and the exponent part $k$ of the final expansion $S$.

### 4.4.3 Square Root Calculation of Mantissa Part

Here, we focus on the mantissa field and propose an accurate calculation algorithm utilizing TSE to satisfy the specified accuracy.

Let us calculate $M$ in [1, 2) using TSE of the square root, for $x$ in [1, 2) to satisfy the given accuracy. TSE of $f(x) = \sqrt{x}$ at center of $a$ is expressed by:

$$f(x) = \sqrt{a} \times \left\{ 1 + \frac{x - a}{2} - \frac{(x - a)^2}{8 \times a} + \frac{(x - a)^3}{16 \times a^2} - \frac{5 \times (x - a)^4}{128 \times a^3} + \frac{7 \times (x - a)^5}{256 \times a^4} - \cdots \right\} \tag{4-11}$$

Fig. 4.1 (a) shows graphs of $f(x) = \sqrt{x}$ and its TSE centered at $a = 1$ where up to 3rd, 4th and 5th terms are taken into account for $1 \leq x < 2$. Their approximation errors are shown in Fig. 4.1 (b). We see from these that the output value accuracy improves, as the number of TSE terms increases.

The approximation error of TSE is defined as follows:

$$\text{Approximation Error} = \left| \frac{f(x) - T_n(x)}{f(x)} \right| \qquad (4\text{-}12)$$

Here, $f(x)$ and $T_n(x)$ are represented by the original function value and the *n*-term TSE value, respectively.

Similarly, Fig. 4.2 shows the result of $f(x) = \sqrt{x}$ utilizing TSE centered at $a = 1.5$.

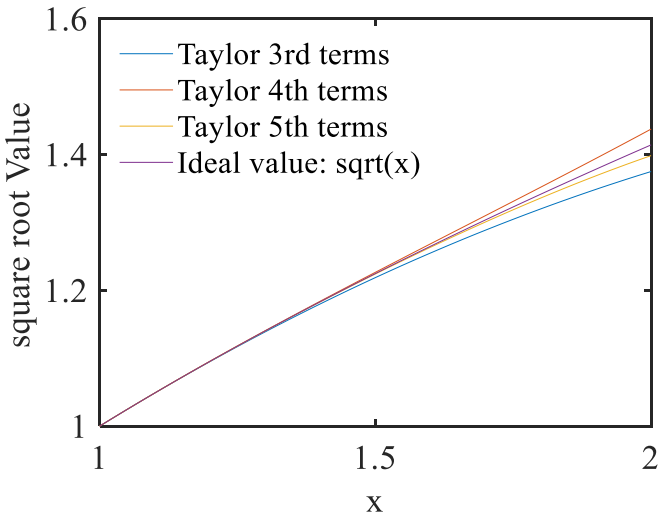
We see from comparison of Figs. 4.1 and 4.2 that the TSE calculation value can be more accurate at the vicinity of the center value of *a*. Thus we use the center point value in the specified range (such as, setting the TSE center point to 1.5 for the range of $[1, 2)$) as the center value *a* of the TSE for better accuracy with good balance. According to this observation, we propose a division technique of the mantissa region in $[1, 2)$ for TSE usage.

Fig. 4.3 shows $f(x) = \sqrt{x}$ for 20-bit accuracy and the convergence range of $f(x)$ TSE with various $a > 0$.

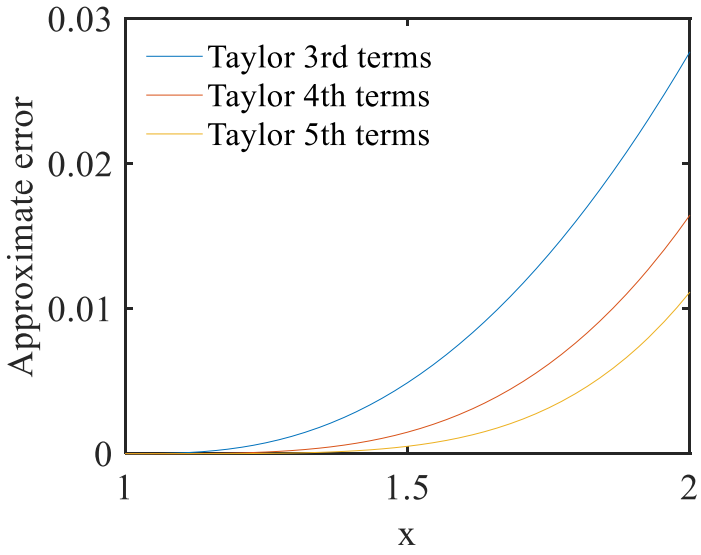
We have conducted simulations to obtain the required number of TSE terms $n$ of $f(x) = \sqrt{x}$ for various cases of required accuracies and specified regions, so as to satisfy the following inequality:

$$max \left| \frac{f(x) - T(n)}{f(x)} \right| \leq r \ . \qquad (4\text{-}13)$$

$f(x)$ is the original function value, $T(n)$ is the *n*-term TSE approximated value, and $r$ is the required accuracy for all $x$ in the specified region.

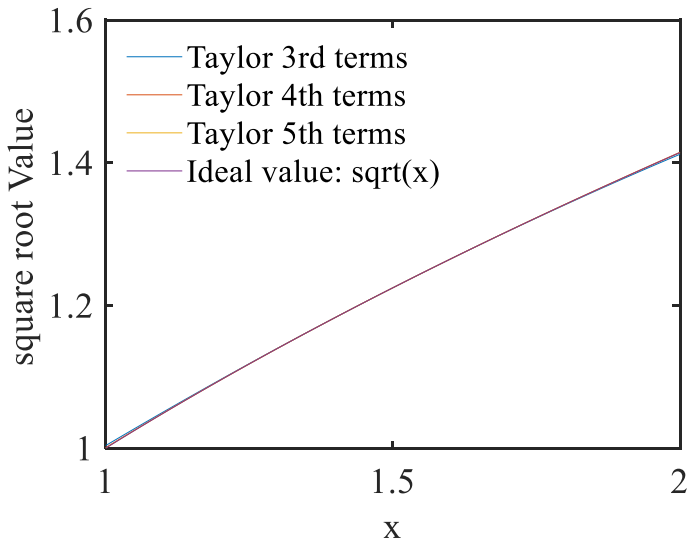



(a) Exact value and TSE value for TSE algorithm of $f(x) = \sqrt{x}$.

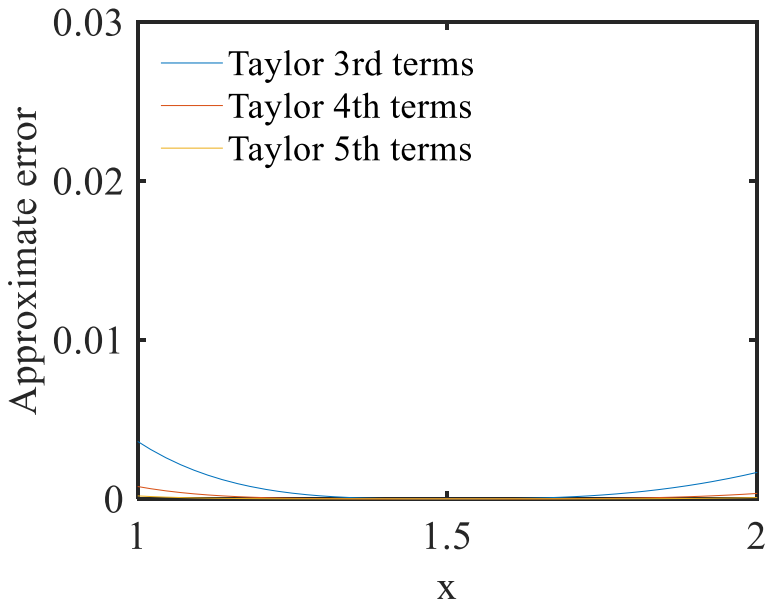



(b) Approximation errors.

Fig. 4.1 TSE with center of $a = 1$.




(a) Exact value and TSE value for TSE algorithm of $f(x) = \sqrt{x}$.




(b) Approximation errors.

Fig. 4.2 TSE with center of $a$=1.5.

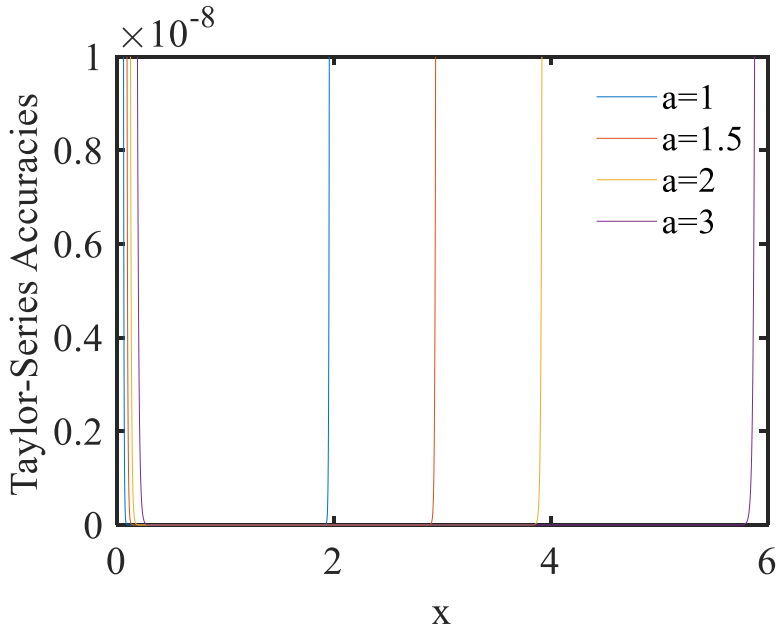



Fig. 4.3 Convergence radius of TSE with various values of $a$ (=1, 1.5, 2, 3) for $f(x) = \sqrt{x}$. Number of TSE terms is 200.

### 4.4.4 Numerical Simulation Results

We have performed numerical simulations to obtain the required number of TSE terms $n$ for $f(x) = \sqrt{x}$ in various cases of required accuracies and specified regions.

(1) One Region of $[1, 2)$

Table 4.1 explains the required number *n* of the TSE terms to meet the specified accuracy or tolerable error, as the simulation result.

D-1-1 TSE of $f(x) = \sqrt{x}$ at $a = 1.5$ for $x$ in $[1, 2)$

When the specified tolerable error $r$ in Eq. (4-13) is $1/2^{16}$ for all $x$ in $[1, 2)$, we obtain $n = 7$ from simulations.

Table 4.1 Required number of TSE terms for specified accuracy with one region of $[1, 2)$.

| Region \ Tolerable error | $\frac{1}{2^8}$ | $\frac{1}{2^{16}}$ | $\frac{1}{2^{20}}$ | $\frac{1}{2^{24}}$ | $\frac{1}{2^{32}}$ |
|---|---|---|---|---|---|
| D-1-1 | 3 | 7 | 9 | 12 | 15 |

(2) Two regions of $[1, 2)$.

We separate the mantissa region of $[1, 2)$ by 2 equally and select the region, according to $M$ (Table 4.2). Then conduct the TSE at the center point $a$ for each region.

Table 4.3 provides simulation results.

Table 4.2 Mantissa regions divided by 2

| | Mantissa Region | Center value $a$ |
|---|---|---|
| D-2-1 | $1 \leq M < 1.5$ | 1.25. |
| D-2-2 | $1.5 \leq M < 2$ | 1.75 |

Table 4.3 Required number of TSE terms for specified accuracy with region of $[1, 2)$ divided by 2.

| **Region** \ **Tolerable error** | $\frac{1}{2^8}$ | $\frac{1}{2^{16}}$ | $\frac{1}{2^{20}}$ | $\frac{1}{2^{24}}$ | $\frac{1}{2^{32}}$ |
|---|---|---|---|---|---|
| D-2-1 | 3 | 5 | 7 | 8 | 11 |
| D-2-2 | 2 | 5 | 6 | 7 | 10 |

(3) Four Regions of $[1, 2)$.

Separate the mantissa region of $[1, 2)$ by 4 equally and select the region, according to *M*. Then conduct TSE at the center point *a* of each divided region (Table 4.4).

Table 4.5 provides our simulation results.

Table 4.4 Mantissa regions divided by 4

| | Mantissa Region | Center value *a* |
|---|---|---|
| D-4-1 | $1 \leq M < 1.25$ | 1.125. |
| D-4-2 | $1.25 \leq M < 1.5$ | 1.375 |
| D-4-3 | $1.5 \leq M < 1.75$ | 1.625 |
| D-4-4 | $1.75 \leq M < 2$ | 1.875 |

Table 4.5 Required number of TSE terms for specified accuracy with region of $[1, 2)$ divided by 4.

| Tolerable error / Region | $\frac{1}{2^8}$ | $\frac{1}{2^{16}}$ | $\frac{1}{2^{20}}$ | $\frac{1}{2^{24}}$ | $\frac{1}{2^{32}}$ |
|---|---|---|---|---|---|
| D-4-1 | 2 | 4 | 5 | 6 | 9 |
| D-4-2 | 2 | 4 | 5 | 6 | 8 |
| D-4-3 | 2 | 4 | 5 | 6 | 8 |
| D-4-4 | 2 | 4 | 5 | 5 | 7 |

(4) Eight Regions of $[1, 2)$

Separate the mantissa region of $[1, 2)$ by 8 equally and select the region, based on $M$. Then conduct TSE at the center point $a$ of each divided region (Table 4.6).

Table 4.7 provides our simulation results.

Table 4.6 Mantissa regions divided by 8

| | Mantissa Region | Center value $a$ |
|---|---|---|
| D-8-1 | $1 \leq M < 1.125$ | 1.0625. |
| D-8-2 | $1.125 \leq M < 1.25$ | 1.1875 |
| D-8-3 | $1.25 \leq M < 1.375$ | 1.3125 |
| D-8-4 | $1.375 \leq M < 1.5$ | 1.4375 |
| D-8-5 | $1.5 \leq M < 1.625$ | 1.5625 |
| D-8-6 | $1.625 \leq M < 1.75$ | 1.6875 |
| D-8-7 | $1.75 \leq M < 1.875$ | 1.8125 |
| D-8-8 | $1.875 \leq M < 2$ | 1.9375 |

Table 4.7 Required number of TSE terms for specified accuracy with region of $[1, 2)$ divided by 8.

| Region \ Tolerable error | $\frac{1}{2^8}$ | $\frac{1}{2^{16}}$ | $\frac{1}{2^{20}}$ | $\frac{1}{2^{24}}$ | $\frac{1}{2^{32}}$ |
|---|---|---|---|---|---|
| D-8-1 | 2 | 3 | 4 | 5 | 7 |
| D-8-2 | 2 | 3 | 4 | 5 | 7 |
| D-8-3 | 2 | 3 | 4 | 5 | 7 |
| D-8-4 | 2 | 3 | 4 | 5 | 6 |
| D-8-5 | 2 | 3 | 4 | 5 | 6 |
| D-8-6 | 2 | 3 | 4 | 5 | 6 |
| D-8-7 | 2 | 3 | 4 | 4 | 6 |
| D-8-8 | 2 | 3 | 4 | 4 | 6 |

### 4.4.5 Region Division for High-Speed SRD Calculation

We investigated to calculate the SRD to meet the specified accuracy with 2 terms of its TSE as follows:

$$f_2(x) = \sqrt{a} \times \left\{1 + \frac{x - a}{2}\right\} = \frac{\sqrt{a}}{2} \times \{x + (2 - a)\} \tag{4-14}$$

Here, $\sqrt{a}\ /2$ and $(2 - a)$ for each divided region are stored in LUT as discussed later. Table 4.8 explains the required number of the mantissa region division for the specified accuracy. The calculation of 2-term TSE requires only 1 multiplication and 1 addition, and hence its calculation can

be done at high speed though the LUT size becomes relatively large.

Table 4.8 Required number of mantissa region division that meets specified accuracy with 2-term TSE.

| Tolerable error | $\frac{1}{2^8}$ | $\frac{1}{2^{16}}$ | $\frac{1}{2^{20}}$ | $\frac{1}{2^{24}}$ | $\frac{1}{2^{32}}$ |
|---|---|---|---|---|---|
| Number of Mantissa Region Division | 4 | 64 | 256 | 1024 | 8192 |

### 4.4.6 Verification with Some Examples

Here, we introduce some examples for comparison between the direct calculation by a digital computer and TSE calculation results for the square root function.

In case of the decimal expression of $\sqrt{X} = \sqrt{1.71875} \times \sqrt{2^8}$, we see that $E$ is an even number, and the following is obtained:

$$S = \sqrt{X} = 1.01001111100111100110_{(2)} \times 2^4$$

After the direct calculation of $\sqrt{1.71875}$ and its conversion to the normalized floating-point format, the following 20-bit mantissa representation is obtained:

$$\sqrt{M} = \sqrt{1.71875} \;=\; 1.01001111100111100110_{(2)} \times 2^0.$$

Here, the proposed TSE method for calculation of $\sqrt{M}$ is used. For instance, in the case of 20-bit accuracy ($\gamma = 1/2^{20}$ in Eq. (4-13)) and $M =$

1.71875 with eight region division, the corresponding region for *M* is shown in D-8-6, and the TSE with 4 terms at *a* = 1.6875 obtained from Eq. (4-11) is given by:

$$t(x) = \sqrt{1.6875} \times \left\{1 + \frac{(x\ - 1.6875)}{2} - \frac{(x\ - 1.6875)^2}{8 \times 1.6875} + \frac{(x\ - 1.6875)^3}{16 \times 1.6875^2}\right\} \quad (4\text{-}15)$$

When $x = 1.71875$, Eq. (4-13) is used and the following is obtained:

$$\text{Taylor} = \ 1.01001111100111100110_{(2)} \times 2^0$$

Their comparison shows that their mantissa parts $1.01001111100111100110_{(2)}$, and the exponent parts (which are 0) of the direct and TSE calculations are the same.

Taking $\sqrt{M} = \ 1.01001111100111100110_{(2)} \times 2^0$ into the odd number of the exponential position, we have the following:

$$S = 1.01001111100111100110_{(2)} \times 2^4$$

Therefore, we find that the mantissa part and the exponent part of $S$ is $1.01001111100111100110_{(2)}$ and $4_{(10)}$ $(= 100_{(2)})$, respectively.

Using the calculation method for *E* being an even number, we can obtain a similar method for *E* being an odd number.

## 4.5 Hardware Design Consideration

Let us consider the hardware implementation complexity for our algorithm to calculate $f(x) = \sqrt{x}$ in various cases.

We investigate the required numbers of multiplications and additions/subtractions using TSE to calculate $S = \sqrt{X} = \sqrt{M} \times \sqrt{2^E}$. For instance, for the 5-term TSE $f_5(x)$ for $f(x) = \sqrt{x}$ at $x = a$, we have the following:

$$f_5(x) = \sqrt{a} \times \left\{1 + \frac{x - a}{2} - \frac{(x - a)^2}{8 \times a} + \frac{(x - a)^3}{16 \times a^2} - \frac{5 \times (x - a)^4}{128 \times a^3}\right\} \tag{4-16}$$

$$= \alpha_0[2 + (x - a)] - \alpha_2(x - a)^2 + \alpha_3(x - a)^3 - \alpha_4(x - a)^4$$

Here:

$$\alpha_0 = \frac{\sqrt{a}}{2}, \alpha_2 = \frac{\sqrt{a}}{8 \times a}, \alpha_3 = \frac{\sqrt{a}}{16 \times a^2}, \alpha_4 = \frac{5\sqrt{a}}{128 \times a^3}.$$

We also define as follows:

$$g_5(x) = \sqrt{2} \times f_5(x)$$

$$= \beta_0[2 + (x - a)] - \beta_2(x - a)^2 + \beta_3(x - a)^3 - \beta_4(x - a)^4 \tag{4-17}$$

Here, $\beta_0 = \sqrt{2}\alpha_0,\ \beta_2 = \sqrt{2}\alpha_2,\ \beta_3 = \sqrt{2}\alpha_3,\ \beta_4 = \sqrt{2}\alpha_4.$

(1) In case that $E$ is even ($E = 2k$):

$S = M_1 \times 2^k$ and $M_1 = f_5(M)$.

(2) In case that $E$ is odd ($E = 2k + 1$):

$S = M_2 \times 2^k$ and $M_2 = \sqrt{2} \times f_5(M) = g_5(M)$.

Here, $a$ is a constant, whereas $x$ is a variable. $\alpha_0, \alpha_2, \alpha_3, \alpha_4$, $\beta_0, \beta_2, \beta_3, \beta_4$ are calculated in advance and written in LUT memory; these are read at calculation time according to on-demand. Then calculate $y = x - a, z = y^2$, which yields:

$$f_5(x) = \alpha_0(2 + y) - z(\alpha_2 - \alpha_3 y + \alpha_4 z) \quad (4\text{-}18)$$

We see that $f_5$ can be calculated by 5 times of floating multiplications and 5 times of additions or subtractions. Table 4.9 explains the necessary numbers of multiplications and additions/subtractions for various numbers of TSE terms for $f(x) = \sqrt{x}$.

Tables 4.10 and 4.9 show that by dividing the mantissa region by 8, the square root of the mantissa can be calculated with 24-bit accuracy by 8 times of multiplications and 8 times of additions or subtractions.

The required LUT size for *N*–divided regions is $N \times 8$ words, and its MSB and 2nd MSB addresses αβ can be obtained directly from *M*=1. αβ. For instance, in the case of the region division of 4 (*N* = 4), the necessary LUT size is 32 words, as shown in Table 4.10.

Table 4.9 Required Amount of Basic Arithmetic Operations for *n*-term TSE.

| Number of TSE terms | Required number of floating multiplications | Required number of floating additions/subtractions |
|---|---|---|
| 3 | 3 | 3 |
| 4 | 4 | 4 |
| 5 | 5 | 5 |
| 6 | 6 | 6 |
| 7 | 7 | 7 |
| 8 | 8 | 8 |

Table 4.10 LUT Contents for 4-Region Division Case

| LUT Address ($\alpha\beta$ $***$) | LUT Data |
|---|---|
| 00 $***$ | $\alpha_0, \alpha_2, \alpha_3, \alpha_4, \beta_0, \beta_2, \beta_3, \beta_4$ for $a$ = 1.125 |
| 01 $***$ | $\alpha_0, \alpha_2, \alpha_3, \alpha_4, \beta_0, \beta_2, \beta_3, \beta_4$ for $a$ = 1.357 |
| 10 $***$ | $\alpha_0, \alpha_2, \alpha_3, \alpha_4, \beta_0, \beta_2, \beta_3, \beta_4$ for $a$ = 1.625 |
| 11$***$ | $\alpha_0, \alpha_2, \alpha_3, \alpha_4, \beta_0, \beta_2, \beta_3, \beta_4$ for $a$ = 1.875 |

## 4.6 Summary

We have exploited the utilization of Taylor-series expansion (TSE) with

uniform mantissa region division to calculate the floating-point square root and clarified quantitatively their implementation hardware design balance among tolerable calculation error, numbers of basic arithmetic operations (multiply, add, subtract) and memory size to satisfy various square root calculation requirements flexibly. The TSE square root calculation method was first mentioned in [11]-[12], but its algorithm design details were not well addressed. Our results allow the designer to build dedicated hardware architectures that suit his/her specific square root calculation requirements.

# References


[1] D. B. Khalifa and M. Martel, "Precision Tuning of An Accelerometer-Based Pedometer Algorithm for IoT Devices," IEEE International Conference on Internet of Things and Intelligence System (IoTaIS), BALI, Indonesia (Jan. 2021).

[2] D. M. Russinoff, “A Mechanically Checked Proof of Correctness Of The AMD K5 Floating-point Square Root Microcode”, Formal Methods in System Design, Springer, vol. 14, pp. 75-125(Jan 1999).

[3] T. Viitanen, P. Jääskeläinen, O. Esko, J. Takala, “Simplified Floating-point Division and Square Root”, IEEE International Conference on Acoustics Speech and Signal Processing, Vancouver, BC, Canada (May 2013).

[4] Y.-C. Liu, Y.-T. Chiang, T.-S. Hsu, C.-J. Liau, D.-W. Wang, “Floating-point Arithmetic Protocols for Constructing Secure Data Analysis Application”, Procedia Computer Science, Vol. 22, pp. 152-161 (Dec. 2013).

[5] I. Sajid, M. M. Ahmed, S. G. Ziavras, “Novel Pipelined Architecture for Efficient Evaluation of the Square Root Using a Modified Non-Restoring Algorithm”, Journal of Signal Processing Systems, Springer, vol. 67, no. 2, pp. 157-166 (May 2012).

[6] J. Wei, A. Kuwana, H. Kobayashi, K. Kubo, "Revisit to Floating-point Division Algorithm Based on Taylor-series Expansion", 16th IEEE Asia Pacific Conference on Circuits and Systems, Ha Long Bay, Vietnam (Dec 2020).

[7] J. Wei, A. Kuwana, H. Kobayashi, K. Kubo, Y. Tanaka, “Floating-point Inverse Square Root Algorithm Based on Taylor-series Expansion”, IEEE Transactions on Circuits and Systems II: Express Briefs, vol. 68, no. 7, pp. 2640-2644 (Feb. 2021).

[8] J. D. Bruguera, “Low Latency Floating-point Division and Square Root Unit”, IEEE Transactions on Computers, vol. 69, no. 2, pp. 274-287 (Feb. 2020).

[9] L. V. Moroz, V. V. Samotyy, O. Y. Horyachyy, “Modified Fast Inverse Square Root and Square Root Approximation Algorithms: The Method of Switching Magic Constants”, MDPI Computation, vol. 9, no. 2, pp. 21 (Feb. 2021).

[10] C. J. Walczyk, L. V. Moroz, J. L. Cieslinski, “Improving the Accuracy of the Fast Inverse Square Root By Modifying Newton–Raphson Corrections”, MDPI Entropy, vol. 23, no. 1, pp. 86 (Jan. 2021).

[11] T.-J. Kwon, J. Sondeen, J. Draper, “Floating-point Division and Square Root Using a Taylor-series Expansion Algorithm”, IEEE Midwest Symposium on Circuits and Systems, Montreal, QC, Canada (Aug. 2007).

[12] O. Gustafsson, L. Wanhammar, “Square Root Computation. Polynomial and Piecewise Polynomial Approximations”, Arithmetic Circuits for DSP Applications (editors, P. K. Meher, T. Stouraitis), IEEE Press, Wiley (2017).

[13] M. Kuhlmann, K. K. Parhi, “Fast Low-Power Shared Division and Square-Root Architecture”, International Conference on Computer Design. VLSI in Computers and Processors, Austin, TX (Oct. 1998).

[14] P. T. P. Tang, “Table Lookup Algorithms for Elementary Functions and Their Error Analysis”, IEEE Symposium on Computer Arithmetic, Grenoble, France (June 1991).

[15] A. Hasnat, T. Bhattacharyya, A. Dey, S. Halder, D. Bhattacharjee, “A Fast FPGA Based Architecture for Computation of Square Root and Inverse Square Root”, Devices for Integrated Circuit, Kalyani, India (Mar. 2017).

[16] A. Raveendran, S. Jean, J. Mervin, D. Vivian, D. Selvakumar, “A Novel Parametrized Fused Division and Square-Root POSIT Arithmetic Architecture”, 33rd International Conference on VLSI Design and 19th International Conference on Embedded Systems, Bangalore, India (Jan. 2020).

[17] T. Bagala, A. Fibich, M. Hagara, P. Kubinec, O. Ondráček, V. Štofanik, R. Stojanović, “Single Clock Square Root Algorithm Based on Binomial Series and Its FPGA Implementation”, 7th Mediterranean Conference on Embedded Computing, Budva, Montenegro (Jun. 2018).

# Chapter 5

# FLOATING-POINT INVERSE SQUARE ROOT ALGORITHM BASED ON TAYLOR-SERIES EXPANSION

## 5.1 Abstract

This chapter describes a segmented structure to deal with inverse square root in floating-point digital calculation arithmetic, based on Taylor-series expansion; it uses only the small number of their expansion terms to achieve a fast evaluation of these functions in high precision. Taylor-series expansions of the inverse square root are examined for several center points with their convergence ranges, and the inverse square root calculation algorithm trade-offs among accuracy, numbers of multiplications/additions/subtractions and LUT sizes are shown; the designer can choose the optimal algorithm for his digital inverse square root calculation, and build its conceptual dedicated hardware architecture design with the contents described here.

Index Terms—Floating-point, division, inverse squares root, Taylor-series expansion

## 5.2 Introduction

With the popularity and rapid development of IoT (internet of things),

our lives have undergone tremendous change. The intelligence level of IoT in industry and life is constantly improving [1]. There the binary floating-point arithmetic plays an important role, and it is widely used in many applications such as signal processing, image processing and scientific computer [2]-[4]. We can understand the floating-point type arithmetic in details from [5].

The inverse square root can be used to improve the accuracy of the second-degree minimax polynomial approximation and Goldschmidt iteration [6]. They proposed an algorithm for elementary function approximation in single-precision floating-point format, which is based on minimax piecewise cubic polynomial approximation and Remes algorithm is used to perform the successive optimization [7], so as to reduce the chip area of LUT and circuit. Digit recurrence algorithm [8] requires less area than other methods, but it has linear convergence and often requires a large number of iterations. In [9], a method is proposed using a table lookup, operand modification, multiplication and Newton-Raphson iteration to reduce delay and chip area. In [10], based on Newton-Raphson method and utilizing fast parallel multipliers, the high performance design can be achieved. But there are also a lot of inverse square root and square root algorithms, which typically either have long latencies or high memory requirements [11]. In many calculations, Cholesky decomposition/Givens rotations/least square lattice filters [12]-[14] and other methods are used to calculate the inverse square root.

There are many operations such as division, multiplication and inverse

square root, using Taylor-series expansion: there the square root or inverse square root algorithm is developed based on the Taylor-series expansion [15]-[16], but there is not much description on the accuracy and convergence range.

The division algorithm based on Taylor-series expansion has been analyzed in [17]. In this chapter, we describe a class of floating-point inverse square root algorithms of our proposed segmented region method for Taylor-series expansion and show their trade-offs among arithmetic accuracy, numbers of multiplications/additions/subtractions and LUT sizes so that the designer can choose the optimal algorithm for his digital inverse square root calculation and dedicated hardware architecture design; its efficient hardware implementation depends on the design target as well as available device technology. We show some numerical calculation results and their hardware implementation considerations, though no specific hardware implementation is not shown.

## 5.3 Inverse Square Root Algorithm

### 5.3.1 Representation of Floating-point Number

A floating-point number consists of three parts, namely the sign bit, the exponent, and the mantissa. Here, we consider a positive number $X$ and ignore the sign bit. Let $X, M, E$, denote the floating-point representation, the mantissa field, the exponent field in binary representation, respectively. So, the binary floating number $X$ can be expressed as:

$$X = M \times 2^E \tag{5-1}$$

Here, the mantissa $M = 1.\alpha\beta\gamma\cdots$ (Here $\alpha, \beta, \gamma\cdots$ is 0 or 1). Notice that the binary point is put so that $1 \leq M < 2$. For example, $M$=1.011001 (binary) = 1.390625 (decimal).

Now let us consider the floating-point algorithm, which calculates the inverse square root of the binary floating-point representation, expressed as $IS$:

$$IS = \frac{1}{\sqrt{X}} = \frac{1}{\sqrt{M}} \times \frac{1}{\sqrt{2^E}} \tag{5-2}$$

### 5.3.2 Exponent Part for Square Root Calculation

Here, we discuss the exponent part by separating the cases that the exponent $E$ is even or odd.

1) When $E$ is even, let $E = 2k$, and the following can be obtained:

$$\frac{1}{\sqrt{2^E}} = 2^{-k} \tag{5-3}$$

$$\frac{1}{\sqrt{X}} = \frac{1}{\sqrt{M}} \times 2^{-k} \tag{5-4}$$

We obtain the exponent part $-$k and the mantissa part $\frac{1}{\sqrt{M}}$ of $IS$. After further normalizing the floating-point type of $\frac{1}{\sqrt{M}}$, the following is obtained:

$$\frac{1}{\sqrt{M}} = \ M_1 \times 2^{k_1} \tag{5-5}$$

Since $1 \leq M < 2$, then $\frac{1}{\sqrt{2}} < M_1 \leq 1$ and hence $k_1 = 0$. We have the following:

$$IS = \frac{1}{\sqrt{X}} = M_1 \times 2^{-k} \tag{5-6}$$

Where, $M_1$ and $-k$ are the mantissa part and the exponent part of $IS$, respectively.

2) When $E$ is odd, let $E = 2k + 1$, and the following is obtained:

$$\frac{1}{\sqrt{2^E}} = \frac{1}{\sqrt{2}} \times 2^{-k} \tag{5-7}$$

After normalizing the floating-point type of $\frac{1}{\sqrt{2}} \times \frac{1}{\sqrt{\mathrm{M}}}$, the following is obtained:

$$\frac{1}{\sqrt{2}} \times \frac{1}{\sqrt{M}} = \ M_2 \times 2^{k_2} \tag{5-8}$$

Since $1 \leq M < 2$, then $\frac{1}{2} < \frac{1}{\sqrt{2}} \times \frac{1}{\sqrt{M}} \leq \frac{1}{\sqrt{2}}$, which leads to $\frac{1}{2} < \ M_2 \leq \frac{1}{\sqrt{2}}$ and $k_2 = 0$, we have the following:

$$IS = \frac{1}{\sqrt{X}} = M_2 \times 2^{-k} \tag{5-9}$$

Where, $M_2$ and $-k$ are the mantissa part and the exponent part of $IS$, respectively.

## 5.4 Taylor-series expansion

Consider an infinitely differentiable function *f(x)*. Its Taylor-series expansion at the center value *a* to obtain the following equation:

$$f(x) = f(a) + f'(a)(x-a) + \frac{(f)''(a)}{2!}(x-a)^2 + \cdots + \frac{(f)^n(a)}{n!}(x-a)^n + \cdots \tag{5-10}$$

Notice that many functions used in engineering design have relatively wide convergence radius; as *n* increases, the Taylor-series expansion reaches $f(x)$ for wide range of *x*.

When $f(x) = 1/\sqrt{x}$ and the center value is *a*, we can obtain the following equation:

$$f(x) = \frac{1}{\sqrt{a}} \times \left\{ 1 - \frac{x-a}{2 \times a} + \frac{3 \times (x-a)^2}{8 \times a^2} - \frac{5 \times (x-a)^3}{16 \times a^3} + \frac{35 \times (x-a)^4}{128 \times a^4} + \cdots \right\} \tag{5-11}$$

### 5.4.1 Taylor-series Expansion Centered at *a*=1

Taylor-series expansion at *a* = 1 can be written as follows:

$$\frac{1}{\sqrt{x}} = 1 - \frac{x-1}{2} + \frac{3 \times (x \ - \ 1)^2}{8} - \frac{5 \times (x \ - \ 1)^3}{16} + \frac{35 \times (x \ - \ 1)^4}{128} + \cdots \quad (5\text{-}12)$$

Figs. 5.1 (a) shows $f(x) = 1/\sqrt{x}$ using Taylor-series expansion centered $a = 1$ when taking 3rd, 4th and 5th terms produce approximated value for ideal inverse square root for $1 \leq x < 2$, noting that the accuracy of the output value increases as number of terms increases.

The inverse square root approximate error in this case of algorithm ($a = 1$) can be expressed in Fig. 5.1(b). The approximate error can be expressed as the following:

$$\text{Approximate Error} = \left| \frac{f(x) - t(n)}{f(x)} \right| \quad (5\text{-}13)$$

Here, $f(x)$ is the ideal value and $t(n)$ is Taylor-series expansion value with *n* terms.

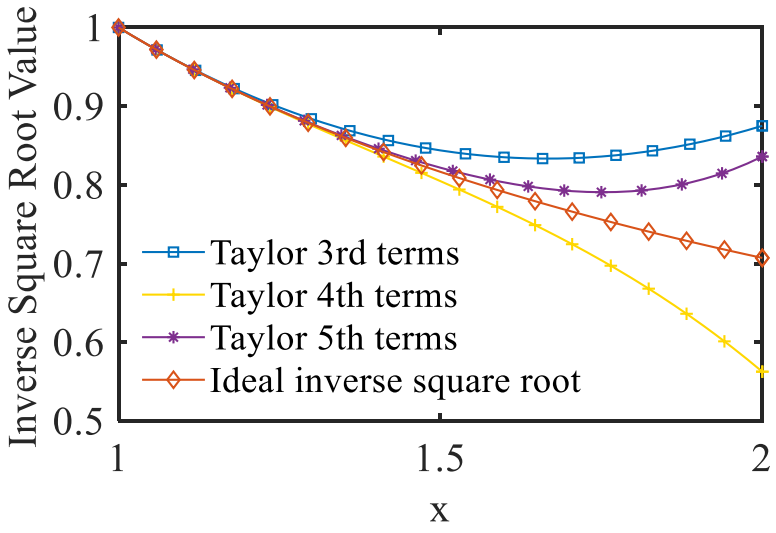
### 5.4.2 Taylor-series Expansion Centered at $a = 1.5$

Taylor-series expansion at *a* = 1.5 can be written as follows:

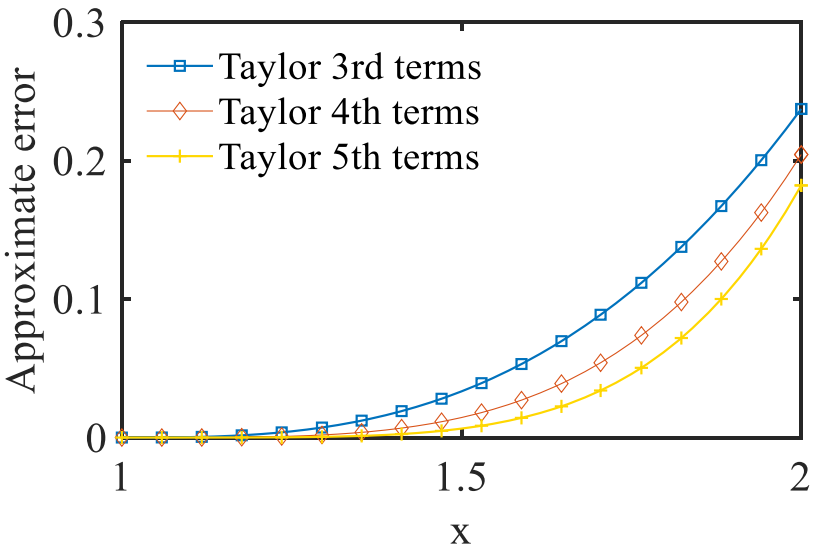
$$\frac{1}{\sqrt{x}} = \frac{1}{\sqrt{1.5}} \times \left\{ 1 - \frac{x-1.5}{2 \times 1.5} + \frac{3 \times (x \ - \ 1.5)^2}{8 \times 1.5^2} - \frac{5 \times (x \ - \ 1.5)^3}{16 \times 1.5^3} + \frac{35 \times (x \ - \ 1.5)^4}{128 \times 1.5^4} + \cdots \right\} \quad (5\text{-}14)$$

Fig. 5.2(a) shows $f(x) = 1/\sqrt{x}$ using Taylor-series expansion centered $a = 1.5$ when taking 3rd, 4th and 5th terms produce approximated value for ideal inverse square root and Fig. 5.2(b) shows approximate error when $1 \leq x < 2$.

By comparing Figs. 5.1 and 5.2, we see that accurate results can be obtained near the center value *a*. So, we use the center point value in the specified range (such as, the center point of 1.5 in the range of $1 \leq x < 2$) as the center value *a* of the Taylor-series expansion for more accuracy. Based on this finding, we propose a segmentation technique of Taylor-series expansion usage.

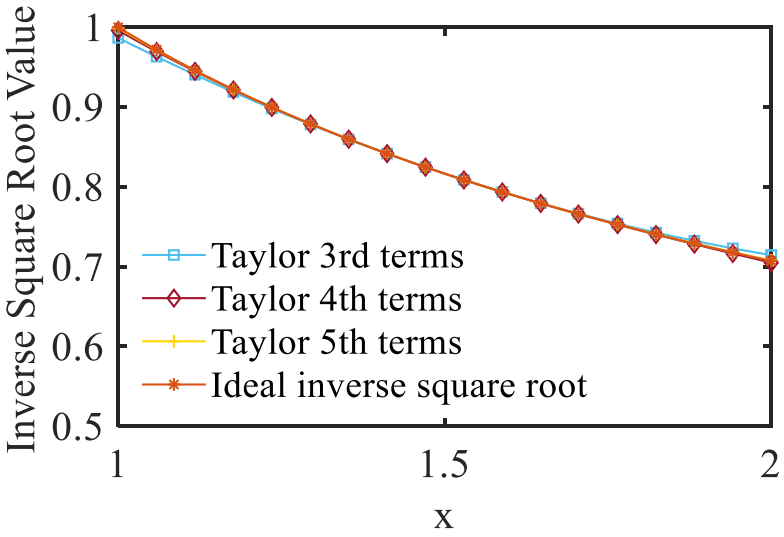



(a) Ideal and Taylor-series expansion values for Taylor-series algorithm of $f(x) = 1/\sqrt{x}$

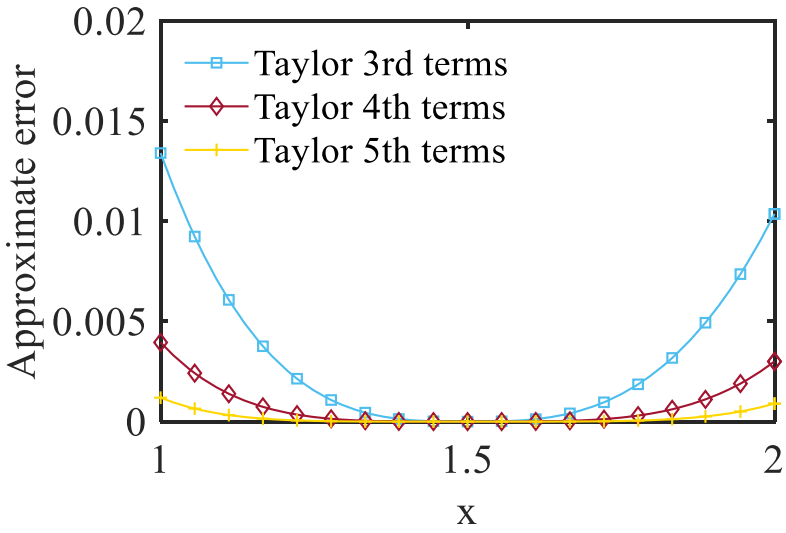



(b) Approximate error of the inverse square root

Fig. 5.1 Taylor-series expansion centered at $a$ =1.




(a) Ideal and Taylor-series expansion values for Taylor-series algorithm of $f(x) = 1/\sqrt{x}$.




(b) Approximate error of the inverse square root.

Fig. 5.2 Taylor-series expansion centered at $a = 1.5$.

## 5.5 Taylor-series Expansion Based Segment Structure Method

In this section, we discuss the method of the segmented structure based on Taylor-series expansion to calculate the floating-point mantissa of the inverse square root.

We have performed simulation to obtain the number of terms $n$ expanded by Taylor-series of $f(x)$ in different cases of given accuracies and given regions, which satisfies the following:

$$\left|\frac{f(x) - t(n)}{f(x)}\right| \leq p \tag{5-15}$$

Where, $f(x)$ is the ideal value, $t(n)$ is Taylor-series expansion value with $n$ terms and $p(x)$ is given accuracy for all $x$ in given region.

Fig. 3 show that $f(x) = 1/\sqrt{x}$ in the accuracy $p$ of $1/2^{20}$ and the convergence range of its Taylor-series expansion with different $a$ $(a > 0)$. As $a$ increases, the converging range also increases, and more accurate value is obtained near the center value. Also we can conjecture the convergence range for various values of $a$. We use the proposed segmentation technique to show the approximate error of 2 and 4 divided regions in Figs. 5.4 and 5.5 respectively; we see that as the number of segments increases, the accuracy also increases.

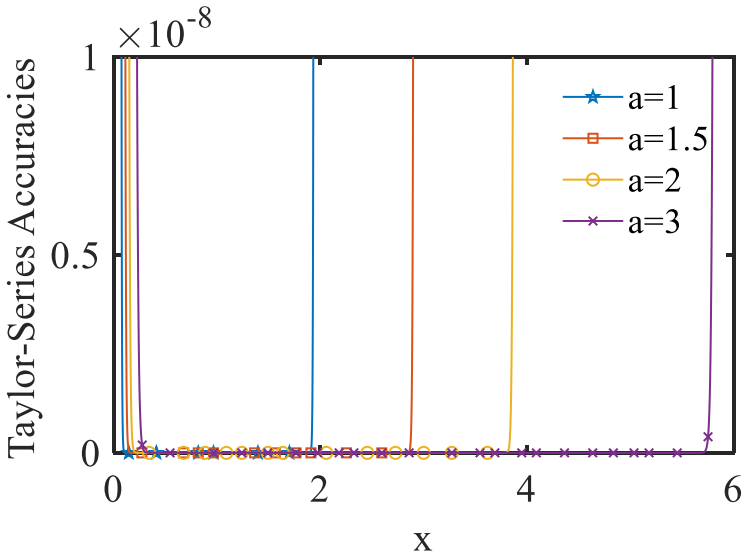
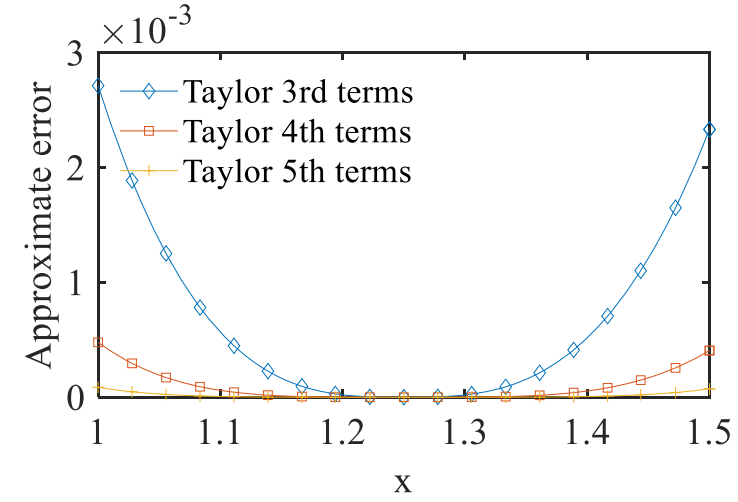
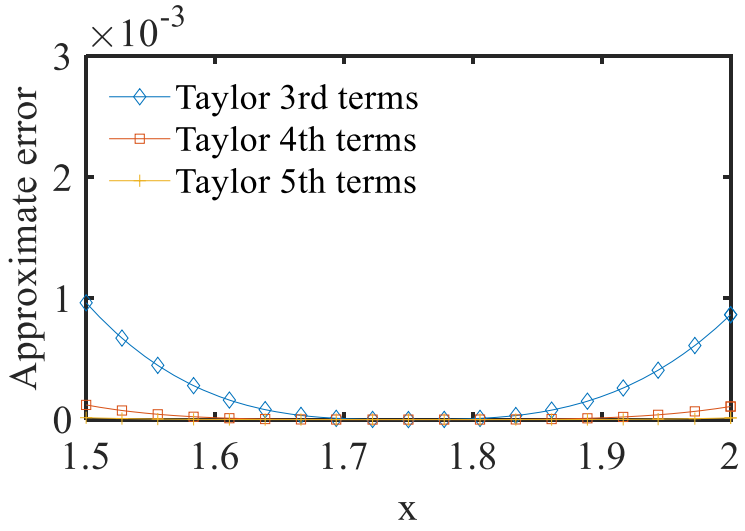



Fig. 5.3 Accuracies and convergence ranges of Taylor-series expansion with various values of $a$ at $f(x) = 1/\sqrt{x}$. Number of terms is 210.

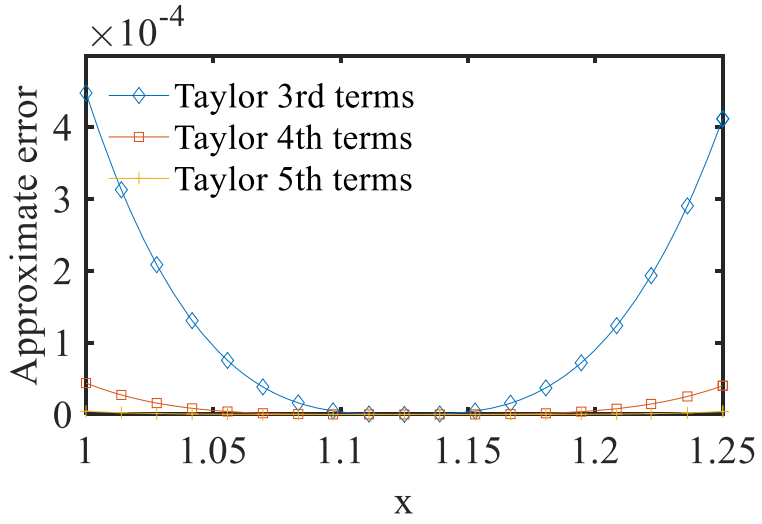



(a) $1 \leq x < 1.5$, $a = 1.25$

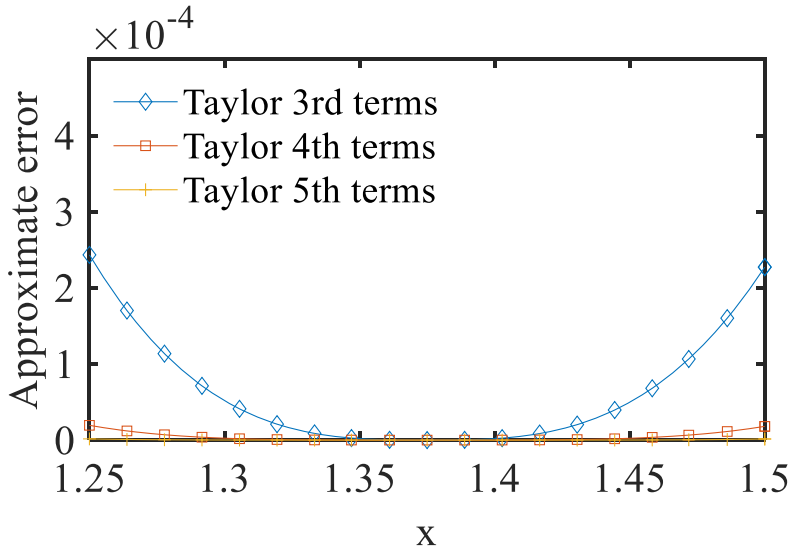



(b) $1.5 \leq x < 2$, $a = 1.75$.

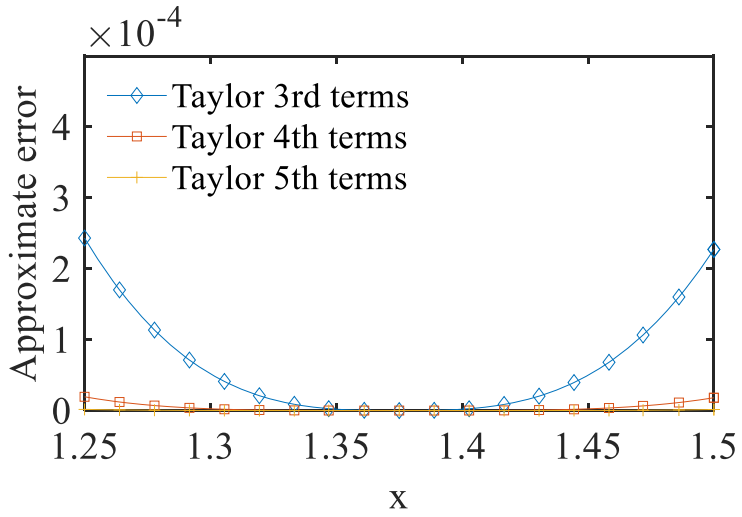
Fig. 5.4 Approximate errors of the proposed segmentation technique in 2 divided regions.



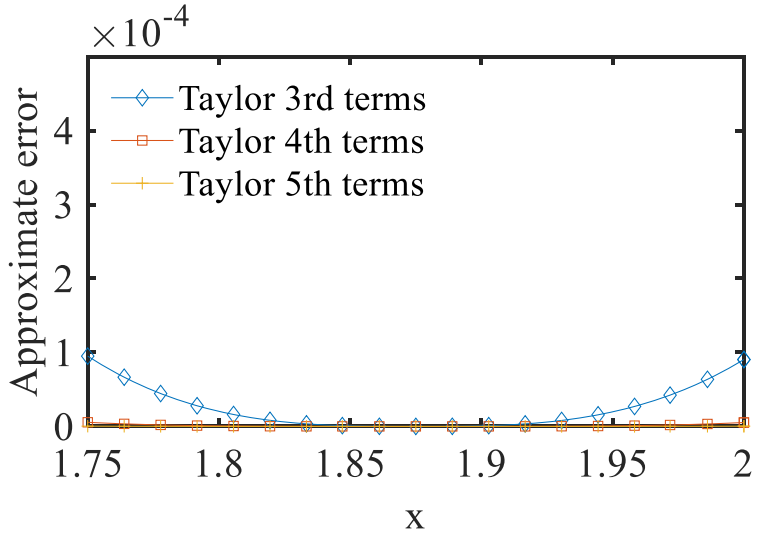

(a) $1 \leq x < 1.25, \quad a = 1.125$




(b) $1.25 \leq x < 1.5$, $a = 1.375$.




(c) $1.5 \leq x < 1.625$, $a = 1.625$.




(d) $1.75 \leq x < 2$, $a = 1.875$.

Fig. 5.5 Approximate errors of the proposed segmentation technique in 4 divided regions.

### 5.5.1 Numerical Simulation Results

1) One Region of $1 \leq x < 2$

Table 5.1 shows the required number of the terms $n$ for Taylor-series expansion to meet the desired accuracy, obtained by numerical simulation.

A-1-a) Taylor-series expansion of $f(x) = 1/\sqrt{x}$ at $a = 1.5$ $(1 \leq x < 2)$

When the given accuracy p in Eq. (5-15) is $1/2^{16}$, for all $x$ $(1 \leq x < 2)$, we obtain $n = 9$ from the numerical simulation.

Table 5.1 Number of Taylor-series expansion terms that meets specified accuracy for one region of $1 \leq x < 2$.

| Precision / Taylor-series expansion | $\frac{1}{2^8}$ | $\frac{1}{2^{16}}$ | $\frac{1}{2^{20}}$ | $\frac{1}{2^{24}}$ | $\frac{1}{2^{32}}$ |
|---|---|---|---|---|---|
| A-1-a) | 5 | 9 | 12 | 14 | 19 |

2) Two regions of $1 \leq x < 2$.

We divide the region of $1 \leq x < 2$ by 2 and choose the region, based on $M$, and perform the Taylor-series expansion at the point $a$ of the center for each region.

A-2-a) For M = 1.0∗∗∗∗ ⋯ $(1 \leq M < 1.5)$, a=1.25.

A-2-b) For M = 1.1∗∗∗∗ ⋯ $(1.5 \leq M < 2)$, a=1.75.

Table 5.2 shows the numerical simulation results.

Table 5.2 Number of Taylor-series expansion terms that meets specified accuracy when region of $1 \leq x < 2$ is divided by 2

| Precision / Taylor-series expansion | $\frac{1}{2^8}$ | $\frac{1}{2^{16}}$ | $\frac{1}{2^{20}}$ | $\frac{1}{2^{24}}$ | $\frac{1}{2^{32}}$ |
|---|---|---|---|---|---|
| A-2-a) | 3 | 7 | 8 | 10 | 13 |
| A-2-b) | 3 | 6 | 7 | 8 | 9 |

3) Divided by 4 of region $1 \leq x < 2$.

Divide the region of $1 \leq x < 2$ by 4 and choose the region, based on *M*. Then perform the Taylor-series expansion at the point *a* of the center of each divided region.

A-3-a) For $M = 1.00{*}{*}{*}{*}\cdots$ $(1 \leq M < 1.25)$, $a = 1.125$.

A-3-b) For $M = 1.01{*}{*}{*}{*}\cdots$ $(1.25 \leq M < 1.5)$, $a=1.375$.

A-3-c) For $M = 1.10{*}{*}{*}{*}\cdots$ $(1.5 \leq M < 1.75)$, $a=1.625$.

A-3-d) For $M = 1.11{*}{*}{*}{*}\cdots$ $(1.75 \leq M < 2)$, $a=1.875$.

Table 5.3 shows the numerical simulation results.

Table 5.3 Number of Taylor-series expansion terms that meets specified accuracy when region of $1 \leq x < 2$ is divided by 4

| Precision / Taylor-series expansion | $\frac{1}{2^8}$ | $\frac{1}{2^{16}}$ | $\frac{1}{2^{20}}$ | $\frac{1}{2^{24}}$ | $\frac{1}{2^{32}}$ |
|---|---|---|---|---|---|
| A-3-a) | 3 | 5 | 6 | 7 | 10 |
| A-3-b) | 2 | 5 | 6 | 7 | 9 |
| A-3-c) | 2 | 4 | 5 | 6 | 8 |
| A-3-d) | 2 | 4 | 5 | 6 | 8 |

4) Divided by 8 of region $1 \leq x < 2$.

Divide the region of $1 \leq x < 2$ by 8 and choose the region, based on $M$. Then perform the Taylor-series expansion at the point $a$ of the center of each divided region.

A-4-a) For *M=1.000∗∗∗∗ ⋯ (*$1 \leq M < 1.125$*), a* =1.0625.

A-4-b) For *M=1.001∗∗∗∗ ⋯(1.125≤ M <1.25), a* =1.1875.

A-4-c) For *M=1.010∗∗∗∗ ⋯* $(1.25 \leq M < 1.375)$*, a* =1.3125.

A-4-d) For *M=1.011∗∗∗∗ ⋯ (*$1.375 \leq M < 1.5$*), a* =1.4375.

A-4-e) For *M=1.100∗∗∗∗ ⋯ (*$1.5 \leq M < 1.625$*), a* =1.5625.

A-4-f) For *M=1.101∗∗∗∗ ⋯* $(1.625 \leq M < 1.75)$*, a* =1.6875.

A-4-g) For *M=1.110∗∗∗∗ ⋯ (*$1.75 \leq M < 1.875$*), a* =1.8125.

A-4-h) For *M=1.110∗∗∗∗ ⋯ (*$1.875 \leq M < 2$*), a* =1.9375.

Table 5.4 shows the numerical simulation results.

Table 5.4 Number of Taylor-series expansion terms that meets specified accuracy when region of $1 \leq x < 2$ is divided by 8

| Precision / Taylor-series expansion | $\frac{1}{2^8}$ | $\frac{1}{2^{16}}$ | $\frac{1}{2^{20}}$ | $\frac{1}{2^{24}}$ | $\frac{1}{2^{32}}$ |
|---|---|---|---|---|---|
| A-4-a) | 2 | 4 | 5 | 6 | 8 |
| A-4-b) | 2 | 4 | 5 | 6 | 7 |
| A-4-c) | 2 | 4 | 5 | 6 | 7 |
| A-4-d) | 2 | 4 | 5 | 5 | 7 |
| A-4-e) | 2 | 4 | 4 | 5 | 7 |
| A-4-f) | 2 | 4 | 4 | 5 | 7 |
| A-4-g) | 2 | 3 | 4 | 5 | 7 |
| A-4-h) | 2 | 3 | 4 | 5 | 7 |

### 5.5.2 Verification with Some Examples

Here, we compare the direct calculation result and the Taylor-series expansion simulation result of the inverse square root calculation with some examples, to verify the proposed method.

In case of the decimal expression $\frac{1}{\sqrt{X}} = \frac{1}{\sqrt{1.71875}} \times \frac{1}{\sqrt{2^8}}$, we see that $E$ is an even number, and the following is obtained:

$$IS = 1/\sqrt{X} = 0.11000011010001001110_{(2)} \times 2^{-4}$$

After the direct calculation of $\frac{1}{\sqrt{1.71875}}$ with a digital processor, and its

normalization to the floating-point type, the following 20-bit representation is obtained:

$$\frac{1}{\sqrt{M}} = \frac{1}{\sqrt{1.71875}} = 0.11000011010001001110_{(2)} \times 2^0$$

Here, we use the proposed Taylor-series expansion method to calculate $1/\sqrt{\mathrm{M}}$. For example, in the case of 20-bit accuracy and $M = 1.71875$ using 8 divided regions, the corresponding case of *M* is in A-4-f, and the Taylor-series with 4 terms at $a$ = 1.6875 (mean: substitute $a$ = 1.6875 into the 4 terms Taylor-series expansion of Eq. (5-11)) is expanded as follows:

$$\frac{1}{\sqrt{x}} = \frac{1}{\sqrt{1.6875}} \times \left\{1 - \frac{x - 1.6875}{2 \times 1.6875} + \frac{3 \times (x - 1.6875)^2}{8 \times 1.6875^2} - \frac{5 \times (x - 1.6875)^3}{16 \times 1.6875^3}\right\} \quad (5\text{-}16)$$

For $x = 1.71875$, the following is obtained:

$$IS = 0.11000011010001001110_{(2)} \times 2^0$$

We see by their comparison that their mantissa parts $0.11000011010001001110_{(2)}$, and exponent parts 0 of the direct and Taylor-series expansion calculations are the same; an error from the ideal value of $\sqrt{1.6875}$ is $3.1922 \times 10^{-8}$ (which is less than $1/2^{20}$) using Eq. (5-13).

Taking $1/\sqrt{M} = 0.11000011010001001110_{(2)} \times 2^0$ into the odd

number of the exponential position, we obtain the following in binary representation:

$$IS = 1/\sqrt{X} = 0.11000011010001001110_{(2)} \times 2^{-4}$$

Therefore, we obtain that the mantissa part and the exponent part of $IS$ is $0.11000011010001001110_{(2)}$ and -4 in binary representation, respectively.

The similar argument holds for odd-number *E* case.

## 5.6 Hardware Implementation Consideration

Let us consider the hardware implementation complexity of using the algorithm under study to perform $f(x) = 1/\sqrt{x}$ calculation in different cases. We consider the required numbers of multiplications/additions/subtractions for Taylor-series expansion to calculate $IS = \frac{1}{\sqrt{X}} = \frac{1}{\sqrt{M}} \times \frac{1}{\sqrt{2^E}}$. For example, in case of 5-term Taylor-series expansion $f_5(x)$ for $f(x) = 1/\sqrt{x}$ at the center value $a$. We have the following:

(1) In case that $E$ is an even number ($E = 2k$):

$IS = M_1 \times 2^{-k}$ and $M_1 = f_5(M) = f_5(x)$.

$$f_5(x) = \frac{1}{\sqrt{a}} \times \left\{ 1 + \frac{x - a}{2 \times a} - \frac{3 \times (x - a)^2}{8 \times a^2} + \frac{5 \times (x - a)^3}{16 \times a^3} - \frac{35 \times (x - a)^4}{128 \times a^4} \right\} \quad (5\text{-}17)$$

$$= \alpha_0 + \alpha_1(x - a) - \alpha_2(x - a)^2 + \alpha_3(x - a)^3 - \alpha_4(x - a)^4$$

Here, $\alpha_0 = \frac{1}{\sqrt{a}}, \alpha_1 = \frac{1}{\sqrt{a^3}}, \alpha_2 = \frac{3}{8 \times \sqrt{a^5}}, \alpha_3 = \frac{5}{16 \times \sqrt{a^7}}, \alpha_4 = \frac{35}{128 \times \sqrt{a^9}}$.

(2) In case that *E* is an odd number ($E = 2k + 1$):

$$IS = M_2 \times 2^{-k} \text{ and } M_2 = \frac{1}{\sqrt{2}} \times f_5(M) = g_5(M) = g_5(x).$$

We also define Eq. (5-18). Here, $\beta_k = \frac{1}{\sqrt{2} \times \alpha_k}$, for $k$ = 0, 1, 2, 3, 4. Also $a$ is a constant and $x$ is a variable. $\alpha_0, \alpha_1, \alpha_2, \alpha_3, \alpha_4$, $\beta_0, \beta_1, \beta_2, \beta_3, \beta_4$ are calculated in advance and stored in LUT memory, which are read at calculation time.

$$g_5(x) = \frac{1}{\sqrt{2}} \times f_5(x) = \beta_0 + \beta_1(x - a) - \beta_2(x - a)^2 + \beta_3(x - a)^3 - \beta_4(x - a)^4 \quad (5\text{-}18)$$

Then we calculate $y = x - a$ and $z = y^2$, we have the following:

$$f_5(x) = \alpha_0 + \alpha_1 y - z(\alpha_2 - \alpha_3 y + \alpha_4 z) \quad (5\text{-}19)$$

We see that $f_5$ can be obtained with 5 multiplications and 5 additions/subtractions. Table 5.5 shows the required numbers of

multiplications and additions/subtractions for the number of Taylor-series terms for $f(x) = 1/\sqrt{x}$. We see from Tables 5.4 and 5.5 that by inverse square root the region by 8, the reciprocal of the mantissa can be calculated with 24-bit accuracy by 5 multiplications and 5 additions/subtractions.

The required LUT size for *N* regions is $N \times 10$ words, and its MSB and 2nd MSB addresses αβ is read for *M*=1. αβ.. Table 5.6 shows the case for *N* = 4 and the LUT size is 40 words.

Table 5.5 Required numbers of multiplications and additions/subtractions for *N*-term Taylor-series expansion

| # of Taylor-series expansion terms | # of multiplications | # of additions/ subtractions | # of LUT words for *N* regions |
|---|---|---|---|
| 3 | 3 | 3 | 8 *N* |
| 4 | 4 | 4 | 10 *N* |
| 5 | 5 | 5 | 12 *N* |
| 6 | 6 | 6 | 14 *N* |
| 7 | 7 | 7 | 16 *N* |
| 8 | 8 | 8 | 18 *N* |

## 5.7 Summary

We have studied floating-point inverse square root algorithms based on Taylor-series expansion using the segmented regions, and shown their hardware implementation trade-offs among simulation accuracy, numbers of

multiplications /additions/subtractions and LUT sizes to meet various digital division specifications flexibly. The original Taylor-series expansion inverse square root calculation method is described in [11], but its design details seem not to be disclosed. We would like to claim that the designer can build his/her conceptual dedicated hardware architecture design for square root calculation with the contents described in this chapter.

Table 5.6 LUT memory for 4 regions ($10 \times 4 = 40$ words)

| Address ($\alpha\beta$ $***$) | LUT data |
|---|---|
| 00 $***$ | $\alpha_0, \alpha_1, \alpha_2, \alpha_3$ , $\alpha_4, \beta_0, \beta_1, \beta_2, \beta_3$ , $\beta_4$ for $a$ = 1.125 |
| 01 $***$ | $\alpha_0, \alpha_1, \alpha_2, \alpha_3$ , $\alpha_4, \beta_0, \beta_1, \beta_2, \beta_3$ , $\beta_4$ for $a$ = 1.357 |
| 10 $***$ | $\alpha_0, \alpha_1, \alpha_2, \alpha_3$ , $\alpha_4, \beta_0, \beta_1, \beta_2, \beta_3$ , $\beta_4$ for $a$ = 1.625 |
| 11$***$ | $\alpha_0, \alpha_1, \alpha_2, \alpha_3$ , $\alpha_4, \beta_0, \beta_1, \beta_2, \beta_3$ , $\beta_4$ for $a$ = 1.875 |

# References


[1] I. Lee, K. Lee, “Internet of Things (IoT): Applications, Investments, and Challenges for Enterprises”, Business Horizons (July 2015).

[2] D. M. Russinoff, “A Mechanically Checked Proof of Correctness of the AMD K5 Floating-point Square Root Microcode”, Formal Methods in System Design, Springer (Jan. 1999).

[3] T. Viitanen, P. Jääskeläinen, O. Esko, J. Takala, “Simplified Floating-point Division and Square Root”, IEEE International Conference on Acoustics Speech and Signal Processing, Vancouver, BC, Canada (May 2013).

[4] Y.-C. Liu, Y.-T. Chiang, T.-S. Hsu, C.-J. Liau, D-W. Wang, “Floating-point Arithmetic Protocols for Constructing Secure Data Analysis Application”, Procedia Computer Science, vol. 22, pp. 152-161 (Oct. 2013).

[5] D. Goldberg, “What Every Computer Scientist Should Know About Floating-point Arithmetic”, ACM Computing Surveys, Vol. 23, No. 1, pp. 5-48 (March 1991).

[6] J. A. Pineiro, J. D. Bruguera, “High-Speed Double-Precision computation of Reciprocal, Division, Square Root, and Inverse Square Root”, IEEE Transactions on Computers, vol. 51, no. 12, pp. 1377-1388 (Dec 2002).

[7] G. J. Cao, H. M. Du, P. C. Wang, Q. Q. Du, J. L. Ding, “A Piecewise Cubic Polynomial Interpolation Algorithm for Approximating Elementary Function”, 14th International Conference on Computer-Aided Design and Computer Graphics, Xi'an, China (Aug. 2015).

[8] S. Majerski, “Square-Rooting Algorithms for High Speed Digital Circuits,” IEEE Transactions on Computer, vol. C-34, no. 8, pp. 724–733 (Aug. 1985).

[9] M. J. Schulte, K. E. Wires, “High-Speed Inverse Square Roots”, 14th IEEE Symposium on Computer Arithmetic, Australia (April 1999).

[10] S. Y. Chen, D. H. Wang, T. J. Zhang, C. H. Hou. “Design and Implementation of a 64/32-bit Floating-point Division, Reciprocal, Square Root, and Inverse Square Root

Unit", 8th International Conference on Solid-State and Integrated Circuit Technology, Shanghai, China (Oct. 2006)

[11] P. Montuschi and P. M. Mezzalama, "Survey of Square Rooting Algorithms", IEE Proceedings E (Computers and Digital Techniques), vol. 137, no. 1, pp. 31–40 (Jan. 1990).

[12] V. K. Jain, G. E. Perez, J. M. Wills, "Novel Reciprocal and Square-Root VLSI Cell: Architecture and Application to Signal Processing", International Conference on Acoustics, Speech and Signal Processing, Toronto, ON, Canada (Apr. 1991).

[13] P. Soderquist, M. Leeser, "Area and Performance Tradeoffs in Floating-Point Divide and Square Root Implementations", ACM Computing Surveys, vol. 28, no. 3, pp. 518–564 (Sept. 1996).

[14] R. W. Stewart, R. Chapman, T. Durrani, "The Square Root in Signal Processing", Proceedings of Real-Time Signal Processing (1989).

[15] M. D. Ercegovac, T. Lang, J. M. Muller, A. Tisserand, "Reciprocation, Square Root, Inverse Square Root, And Some Elementary Functions Using Small Multipliers", IEEE Transactions on Computers, vol. 49, no.7, pp. 628–637 (Jul. 2000)

[16] T. J. Kwon, J. Draper, "Floating-point Division and Square Root using a Taylor-series Expansion Algorithm", Microelectronics Journal, vol. 40, no. 11, pp. 1601–1605 (Nov. 2009).

[17] J. Wei, A. Kuwana, H. Kobayashi, K. Kubo, "Revisit to Floating-Point Division Algorithm Based on Taylor-series Expansion", IEEE Asia Pacific Conference on Circuits and Systems, Ha Long, Vietnam (Dec. 2020).

# Chapter 6

# DIVIDE AND CONQUER: FLOATING-POINT EXPONENTIAL CALCULATION BASED ON TAYLOR-SERIES EXPANSION

## 6.1 Abstract

This paper presents an algorithm to compute the exponential $exp(x)$ floating-point tails based on Taylor-series expansion with mantissa region division. $exp(x)$ is expanded in different regions with corresponding central values using Taylor-series and the best result is selected from among the different convergence ranges obtained. We show the cases of $x>0$ as well as $x<0$, and then show the tradeoff among LUT size and the required numbers of additions, subtractions and multiplications, and also computing accuracy of $exp(x)$ by Taylor-series expansion through simulation results. The designer can choose the best algorithm to build a reasonable hardware system by the method described in this chapter.

Keywords - Floating-point, Exponential, Taylor-series Expansion, Digital Arithmetic, Divide and Conquer.

## 6.2 Introduction

As VLSI integration continues to advance and technical requirements become more demanding, binary floating-point computing becomes more

important, and we must satisfy the need for high precision and real-time floating-point calculations in a wider range of applications. There has been a lot of research on basic operations (division) and inverse squares root [1-2]. Here, this chapter focuses on floating-point index calculation.

The exponential function has a very broad application area including data analytics, simulation of neural networks, web search, and Fourier transform. A report released by Microsoft research group showed that commonly used operations include division, logarithm and exponential, with wide variety of scientific applications, for which latency, accuracy, chip area and power are important requirements [3]. There are various solutions to these problems and also many studies have addressed exponential function calculation algorithms. It is described in [4] how exponentiation can be approximated by manipulating the components of the standard (IEEE-754 in Fig. 1) floating-point representation. In reference [5], a generalized hyperbolic coordinate rotating digital computer (GH CORDIC) is proposed that allows direct computation of numbers and indices. Reference [6] uses Newton's method to solve the numerical instability caused by the singularity generated by using exponential integrators in transient circuit simulations.

Taylor-series expansion is used for exponential calculation in [7]. There are many operations that use Taylor-series expansion for floating-point calculations. However, there is not much discussion on the degree of convergence range and accuracy of Taylor-series expansion. As an adjunct to previous work [7], this paper describes an algorithm for computing floating-point mantissa region division using Taylor-series expansion. We

show that as the number of the divided regions for the mantissa increases, the required numbers of multiplications/additions/ subtractions are reduced; a specified accuracy can be achieved at the cost of an increase in LUT size. We investigate the cases of $x<0$ and $x>0$ and find that their calculation complexities are almost the same.

In this chapter, we also elucidate trade-offs between LUT size, arithmetic precision and usage of basic arithmetic operations: addition, subtraction and multiplication. A designer can build efficient hardware devices by the method proposed in this chapter. Hardware design as implemented depends on the design criteria and available hardware elements.

## 6.3 Investigated exponential algorithm

### 6.3.1 Representation of floating-point number

The IEEE-754 format is a floating-point number that generally consists of the sign bit part *S*, the exponent part *E*, and the mantissa part *M* as shown in Fig. 6.1.

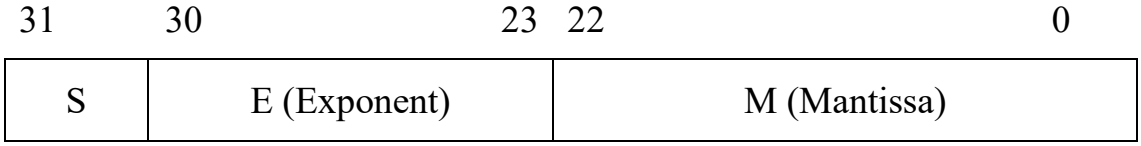



Fig. 6.1 IEEE-754 single-precision floating-point format.

The general IEEE-754 standard can consist of double-precision 64-bit, single-precision 32-bit and half-precision 16-bit; precision can be selected by the designer.

The discussion here is mainly based on the sign bit being positive. The binary floating-point number *X* consists of an exponential part and a mantissa part, denoted as *E* and *M*, respectively, as shown by:

$$\mathrm{X} = \mathrm{M} \times 2^{\mathrm{E}} \tag{6-1}$$

Here, the floating-point mantissa is $M = 1.\mathrm{ABC}\cdots$ (Here, $\mathrm{A}, \mathrm{B}, \mathrm{C}\cdots$ is 1 or 0), and its range is $1 \leq M < 2$. For example, $M$ = 1.01111 (binary) = 1.46875 (decimal).

Now let us consider the floating-point algorithm that calculates the exponential of the binary floating-point representation, expressed as $EXP$:

$$EXP = exp(M \times 2^{E}) = (exp(M))^{2^{E}} \tag{6-2}$$

In the following sections, we discuss how to calculate the mantissa of $exp(M)$.

### 6.3.2 Exponential Calculation by Taylor-series Expansion

Taylor-series expansion can be expressed by the expansion of a function *f(x)* infinitely differentiable at *x* = *a*, which can be written as the following expression:

$$f(x) = f(a) + f'(a)(x-a) + \frac{(f)''(a)}{2!}(x-a)^2 + \cdots + \frac{(f)^n(a)}{n!}(x-a)^n + \cdots \tag{6-3}$$

Consider the calculation of the exponential $exp(M)$ of the floating-

point mantissa $M$ ($1 \leq M < 2$). Let the central value $x = a$ ($1 \leq a < 2$), $f(x) = exp(x)$ be computed using Taylor-series expansion expressed as follows:

$$\begin{aligned} f(x) &= exp(a)\,[1 + q + \frac{q^2}{2} + \frac{q^3}{6} + \frac{q^4}{24} + \frac{q^5}{120} + \frac{q^6}{720}] \\ &= exp(a)[1 + q(1 + q(\frac{1}{2} + q(\frac{1}{6} + \frac{q}{24} + q(\frac{1}{120} + \frac{q}{720}))))] \end{aligned} \tag{6-4}$$

Taylor-series expansion of the exponential approximation up-to the 6th term is shown in Eq. (6-4), where $q = x - a$. This is a well-known method to reduce the number of multiplications in Taylor-series [7].

Fig. 6.2 (a) shows the approximate results of $f(x) = exp(x)$ by Taylor-series expansion with 3rd, 4th and 5th terms at the central value $a = 1$; it also shows the ideal value of $f(x) = exp(x)$ (range: $1 \leq x < 2$). We see in Fig. 6.2 that as the number of terms increases, the approximation approaches the ideal value.

Fig. 6.2 shows the approximation error (AE) between the ideal and Taylor-series expansion, the approximation error is defined as follows:

$$\text{AE} = \left| \frac{f(x) - t_n(x)}{f(x)} \right| \tag{6-5}$$

$t_n(x)$ and $f(x)$ represent the Taylor-series expansion approximation value with $n$-term and ideal value, respectively.

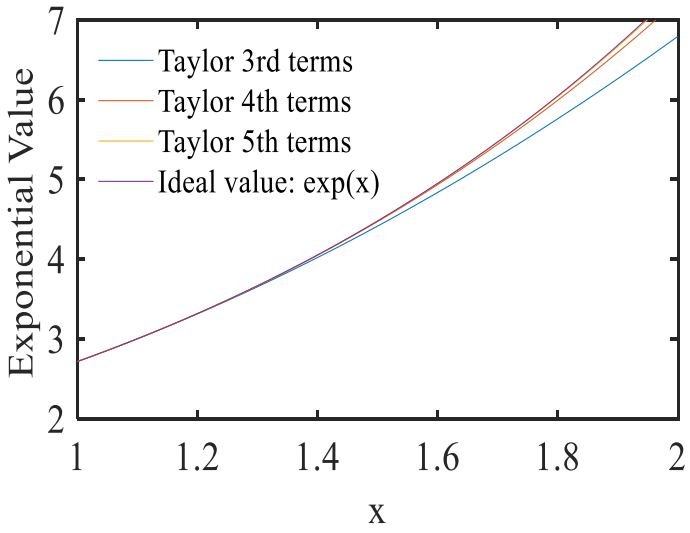
Similarly, Fig. 6.3(a) and 6.3(b) show the approximation results of the Taylor-series expansion of $f(x) = exp(x)$ when the center value $a = 1.5$.

With the same number of Taylor-series expansion terms and the same interval $(1 \leq x < 2)$, all the results shown in Fig. 6.3 are close to the ideal values of Fig. 6.2. That is, the values around the central value *a* are close to the ideal values. Therefore, it can be concluded that between the specified range $1 \leq x < 2$, taking the center point of the range as the central value $a$ of Taylor-series expansion can yield accurate results. With this finding, a technique of region division for the exponential calculation of the mantissa using Taylor-series expansion is proposed.

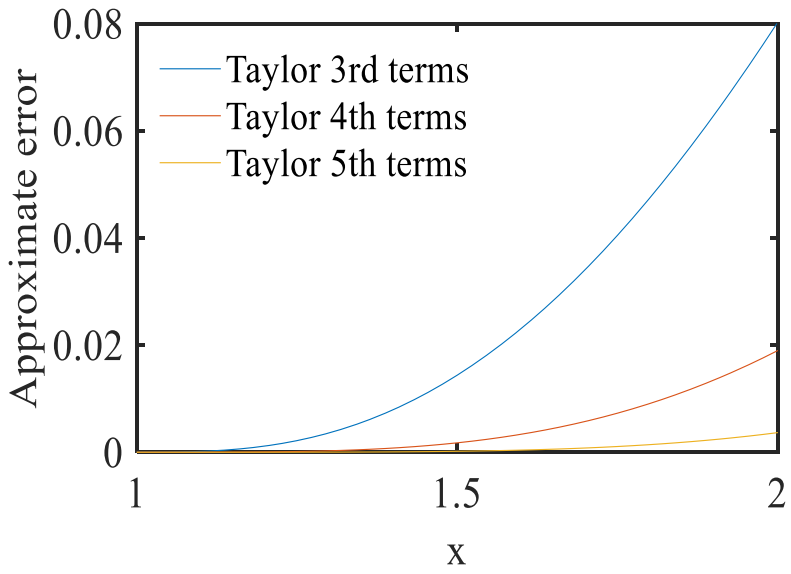
## 6.4 Simulation results

In this section, our mantissa region division method for the region of $1 \leq x < 2$ based on Taylor-series expansion is used to calculate the mantissa of the exponential calculation. Based on the method proposed in this paper, with the specified region and accuracy *p*, $f(x) = exp(x)$ using the minimum number of the terms *n* needed for Taylor-series expansion, the following equation can be derived:

$$max\left|\frac{f(x) - t_n(x)}{f(x)}\right| \leq p \tag{6-6}$$

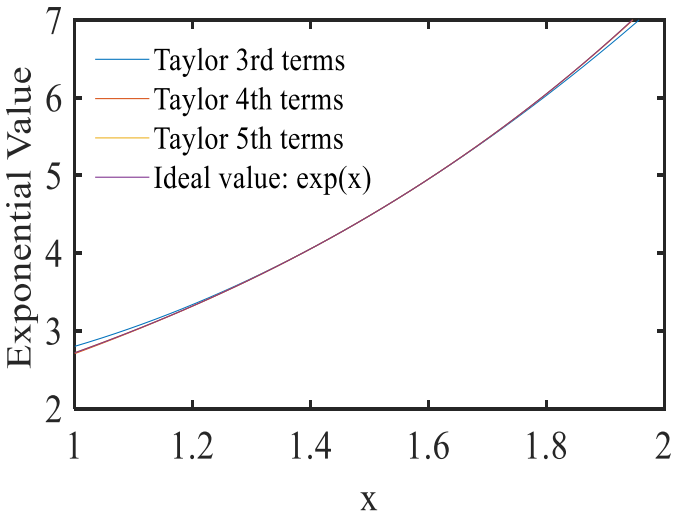



(a) Approximation value and ideal value.

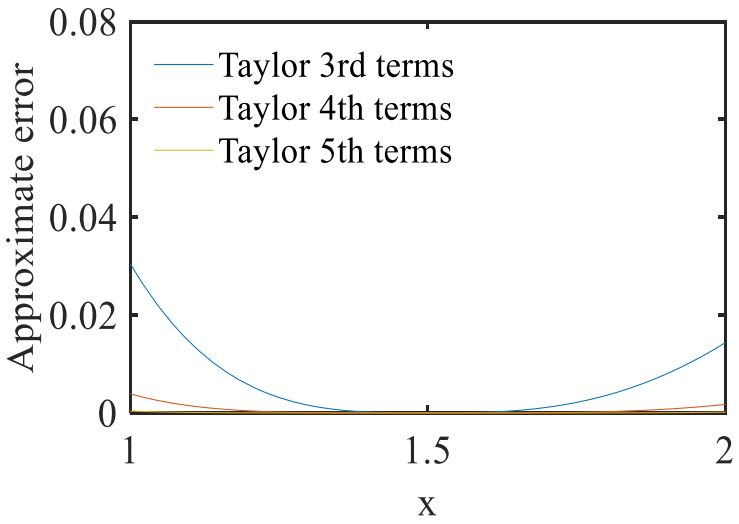



(b) Taylor-series expansion AE.

Fig. 6.2 At the central value $a = 1$, the result of $f(x) = \exp(x)$ using Taylor-series expansion.




(a) Approximation value and ideal value

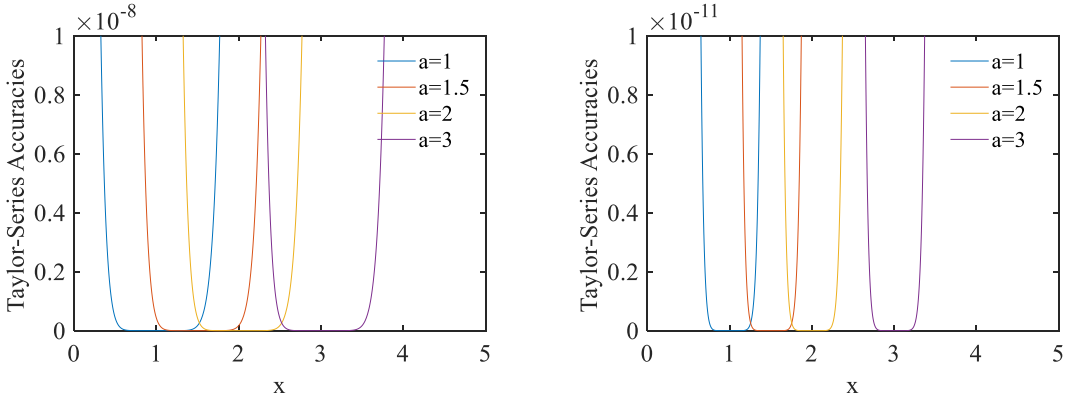



(b) Taylor-series expansion AE.

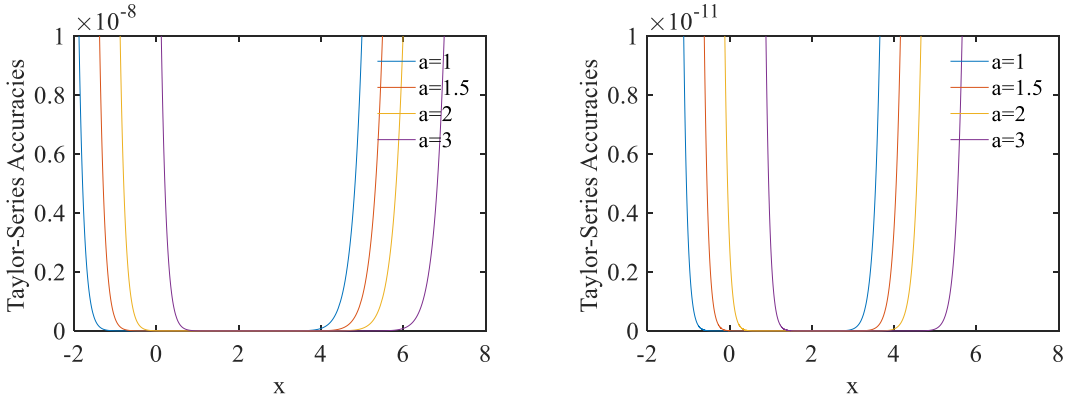
Fig. 6.3 At the central value $a = 1.5$, the result of $f(x) = exp(x)$ using the Taylor-series expansion.

Fig. 6.4 shows the convergence range of different cases under 20-bit and 32-bit precision using Taylor-series expansion $f(x) = exp(x)$. The left

sides of Figs. 6.4 (a) and 6.4 (b) show 20-bit precision, and their right sides show 32-bit precision. Comparative analysis shows that the convergence range lies around the center point, and as the accuracy improves, the convergence range narrows. Also, as the number of Taylor-series expansion items increases, the convergence range also increases.




(a) The number of Taylor-series expansion terms is 10, with different central values.




(b) The number of Taylor-series expansion terms is 20, with different central values.

Fig. 6.4 $f(x) = exp(x)$ Taylor-series expansion in different cases.

### 6.4.1 Binary point position of mantissa $M = 1.\ ABC\cdots\ (1 \le M < 2)$

(a) Non-divided Mantissa Region Case

Table 6.1 shows the exponents of the tails calculated using Taylor-series expansions; the minimum number of terms *n* of the Taylor-series expansion required to meet the specified accuracy is obtained by simulation.

(1) Taylor-series expansion of $f(x) = exp(x)$ at $a = 1.5$ $(1 \le x < 2)$

We see from Table I that the minimum number of terms for Taylor-series expansion is 8 for the accuracy of $1/2^{20}$ by Eq. (6-6).

Table 6.1 Precision and number of Taylor-series terms in one region of $1 \le x < 2$.

| Accuracy | | $\frac{1}{2^8}$ | $\frac{1}{2^{16}}$ | $\frac{1}{2^{20}}$ | $\frac{1}{2^{24}}$ | $\frac{1}{2^{32}}$ |
|---|---|---|---|---|---|---|
| (1) | Terms (*n*) | 4 | 7 | 8 | 9 | 11 |

(b) Mantissa Region Divided by 2

Table 6.2 shows that $1 \le x < 2$ is divided into two regions, the appropriate region is chosen according to the size of the tail and a Taylor-series expansion is then performed based on the chosen region. Table 6.3 shows the calculation results.

Table 6.2 Dividing $1 \leq x < 2$ into 2 regions

| Region | Center value *a* | Mantissa value |
|---|---|---|
| (1) | 1.25 | $1 \leq x < 1.5$ |
| (2) | 1.75 | $1.5 \leq x < 2$ |

Table 6.3 Precision and number of Taylor-series terms in 2-region division of $1 \leq x < 2$.

| Accuracy | | $\frac{1}{2^8}$ | $\frac{1}{2^{16}}$ | $\frac{1}{2^{20}}$ | $\frac{1}{2^{24}}$ | $\frac{1}{2^{32}}$ |
|---|---|---|---|---|---|---|
| (1) | Terms (*n*) | 3 | 5 | 6 | 7 | 9 |
| (2) | | 3 | 5 | 6 | 7 | 9 |

(c) Mantissa Region Divided by 4

The same approach divides 1< *x* <2 into 4 regions, Table 6.4 shows the region format and Table 6.5 shows the calculation results.

Table 6.4 Dividing $1 \leq x < 2$ into 4 regions.

| Region | Center value *a* | Mantissa value |
|---|---|---|
| (1) | 1.125 | $1 \leq x < 1.25$ |
| (2) | 1.375 | $1.25 \leq x < 1.5$ |
| (3) | 1.625 | $1.5 \leq x < 1.75$ |
| (4) | 1.875 | $1.75 \leq x < 2$ |

Table 6.5 Precision and number of Taylor-series terms in 4-region division of $1 \leq x < 2$.

| Accuracy | | $\frac{1}{2^8}$ | $\frac{1}{2^{16}}$ | $\frac{1}{2^{20}}$ | $\frac{1}{2^{24}}$ | $\frac{1}{2^{32}}$ |
|---|---|---|---|---|---|---|
| (1) | Terms ($n$) | 3 | 4 | 5 | 6 | 7 |
| (2) | | 3 | 4 | 5 | 6 | 7 |
| (3) | | 3 | 4 | 5 | 6 | 7 |
| (4) | | 3 | 4 | 5 | 6 | 7 |

Similarly, it is possible to divide $1< x <2$ into higher multiples of regions, such as 8, 16, 32, and so on.

### 6.4.2 Verification with Some Examples

In the following, the exponential calculations are compared by Taylor-series expansions and direct calculations.

As the first example, we calculate the binary number 1011001, where the floating-point representation type is $x = 1.011001 \times 2^6$ (binary) $= 1.390625 \times 64$ (decimal). The exponential for calculation $x$ can be obtained as follows:

$$exp(x) = exp(1.390625 \times 64)$$

$$= (exp(1.390625))^{64}$$

Here, we calculate the mantissa of $exp(1.390625)$ and verify the proposed method; MATLAB simulation is the direct method used to obtain

double precision results:

$$exp(1.390625) = 4.017360118591115$$

We compute the exponent $exp(M)$ of the floating-point mantissa ($M$=1.139062) by the proposed method. For example, consider the case that the computation accuracy of $1/2^{16}$ needs to be achieved. It is possible to divide $1 \leq x < 2$ into four regions combining Tables 6.4 and 6.5, which gives the central value $a$ = 1.375 and the number of Taylor-series expansion terms $n$ = 5; expanding the expressions yields:

$$f(x) = exp(1.375) \times \left\{1 + (x - 1.375) + \frac{(x - 1.375)^2}{2} + \frac{(x - 1.375)^3}{6} + \frac{(x - 1.375)^4}{24}\right\} \quad (6\text{-}7)$$

For $x = 1.390625$, we can obtain the approximate value of 4.017360108737799 using Eq. (6-7) and the error from the ideal value of $exp(1.390625)$ is $2.452684 \times 10^{-9}$ (which is less than $1/2^{16}$) using Eq. (6-6).

## 6.5 Exponential calculation for negative number

Now let us consider the calculation of $f(x) = exp(x)$ for $x < 0$, where the sign bit indicates "minus" and $1 \leq M < 2$ in Fig. 6.1. The accurate calculation of $f(x) = exp(x)$ for $x < 0$ is often required in assessing transient phenomena of electric circuits; $exp(-t/T)$ has to be calculated with a time constant of $T$.

In this case, we also performed simulations and calculations, and the results obtained are exactly the same as for the positive mantissa $(1 \leq x < 2)$, except for the case shown in Table 6.6 (accuracy: $1/2^{32}$ ). See Table 6.1. The results obtained in Tables 6.3 and 6.5 can be used for $-2 < x \leq -1$ in the cases of region division by 2 and 4, respectively.

Similarly, we found that the calculation complexity of $f(x) = exp(x)$ for $0 \leq x < 1$ is the same as for $1 \leq x < 2$.

Table 6.6 Precision and number of Taylor-series terms in one region of $-2 < x \leq -1$.

| Accuracy | | $\frac{1}{2^8}$ | $\frac{1}{2^{16}}$ | $\frac{1}{2^{20}}$ | $\frac{1}{2^{24}}$ | $\frac{1}{2^{32}}$ |
|---|---|---|---|---|---|---|
| One region | Terms (n) | 4 | 7 | 8 | 9 | 10 |

## 6.6 Hardware Implementation Consideration

The hardware complexity can be derived from the calculations below. Taylor-series expansion term n=5 is used as an example to calculate the LUT size and the numbers of additions, subtractions and multiplications:

$$f_5 = exp(a)\left\{1 + (x-a) + \frac{(x-a)^2}{2} + \frac{(x-a)^3}{6} + \frac{(x-a)^4}{24}\right\} \qquad (6\text{-}8)$$

Where *a* and *x* denote a constant and a variable, respectively. $exp(a)$ values are first stored in the LUT. Let $y = x\text{-}a$, $z = y^2$. Eq. (6-8) can be turned into the following equation:

$$f_5 = exp(a) \times (1 + y + \frac{y^2}{2} + \frac{y^3}{6} + \frac{y^4}{24}) \\ = exp(a) \times \left\{1 + y + \frac{z}{2} \times (1 + \frac{y}{3} + \frac{z}{12})\right\} \quad (6\text{-}9)$$

The expansion of the 5-term number can be calculated to be composed of 6 multiplications and 5 additions and subtractions. We know from Table 6.5 that the expansion of the 5-term number can reach an accuracy of $1/2^{20}$. Table 6.7 shows, using the same calculation as above, the relationship between the *n*-term and the number of additions, subtractions and multiplications of $f(x) = exp(x)$ when using Taylor-series expansion.

Table 6.7 Numbers of additions, subtractions and multiplications required for *n*-term expansion.

| # of Taylor-series expansion terms ($n$) | # of multiplications | # of additions and subtractions |
|---|---|---|
| 3 | 3 | 3 |
| 4 | 5 | 4 |
| 5 | 6 | 5 |
| 6 | 8 | 6 |
| 7 | 9 | 7 |
| 8 | 10 | 8 |

To calculate LUT size, the value of the calculated $exp(a)$ is first stored in memory ready to be read. Table 6.8 shows that the mantissa is divided into 4 regions requiring 4 words, and so on, and other regions can be

used to calculate the LUT size in the same way; the data at address αβ is read for $M = 1$. αβ….

Note that as the number of the required terms for Taylor-series expansion for a specified accuracy is reduced thanks to our region division method, the number of LUT accesses is also decreased.

Table 6.8 Mantissa divided into 4 regions in LUT memory.

| Address ($\boldsymbol{\alpha\beta}$) | LUT data |
|---|---|
| 00 | *Exp(a)* for $a = 1.125$ |
| 01 | *Exp(a*) for $a = 1.357$ |
| 10 | *Exp(a)* for $a = 1.625$ |
| 11 | *Exp(a)* for $a = 1.875$ |

## 6.7 Summary

Several papers have already described Taylor-series expansion methods for floating-point calculations, but they fail to provide detailed discussions of precision and the number of Taylor-series expansion terms required. Based on the computation of floating-point exponents using Taylor-series expansions proposed in [7], this paper has discussed the required convergence range and precision, focusing on the number of terms and the center value for Taylor-series expansion. This paper elucidated the trade-offs between LUT size, and accuracy, and the basic arithmetic that engineers can use to build optimal digital algorithms based on the design concepts

presented in this paper.

We conclude this chapter by emphasizing that previous works such as [7] used the Taylor-series expansion for the exponential calculation without region division. We described a region division method specifically intended to reduce the number of calculations as the number of the region division increases: equivalently, with the same number of the terms. compared with the original method in [7], our method can greatly improve the accuracy.

# References


[1] J. Wei, A. Kuwana, H. Kobayashi and K. Kubo, “Revisit to Floating-point Division Algorithm Based on Taylor-series Expansion”, IEEE Asia Pacific Conference on Circuits and Systems, Ha Long, Vietnam (Dec. 2020).

[2] J. Wei, A. Kuwana, H. Kobayashi, K. Kubo and Y. Tanaka, “Floating-point Inverse Square Root Algorithm Based on Taylor-series Expansion”, IEEE Transactions on Circuits and Systems II: Express Briefs, vol. 68, no. 7, pp. 2640-2644 (Feb. 2021).

[3] J. Fowers, K. Ovtcharov, K. Strauss, E. S. Chung, and G. Stitt, “A High Memory Bandwidth FPGA Accelerator for Sparse Matrix-Vector Multiplication”, IEEE 22nd International Symposium on Field-Programmable Custom Computing Machines, Boston, MA (May. 2014).

[4] N. N. Schraudolph, “A Fast, Compact Approximation of The Exponential Function”, Neural Computation, vol. 11, no.4, pp. 853–862 (Jun. 1991).

[5] Y. Luo, Y. Wang, Y. Ha, Z. Wang, S. Chen and H. Pan, "Generalized Hyperbolic CORDIC and Its Logarithmic and Exponential Computation with Arbitrary Fixed Base", IEEE Transactions on Very Large Scale Integration Systems, vol. 27, no. 9, pp. 2156-2169 (Jun. 2019).

[6] Q. Chen, “A Robust Exponential Integrator Method for Generic Nonlinear Circuit Simulation”, 57th ACM/IEEE Design Automation Conference, San Francisco, CA (Jul. 2020).

[7] P. Nilsson, A. U. R. Shaik, R. Gangarajaiah, E. Hertz, “Hardware Implementation of the Exponential Function using Taylor-series”, IEEE NORCHIP Conference, Tampere, Finland (Oct. 2014).

# Chapter 7

# FLOATING-POINT LOGARITHMIC ALGORITHMS BASED ON TAYLOR-SERIES EXPANSION WITH MANTISSA REGION DIVISION AND CONVERSION

## 7.1 Abstract

This paper investigates floating-point calculation algorithms for logarithm with base 2, $f(x) = \log_2 x$, based on Taylor-series expansion. Three techniques are investigated; (i) The first one is uniform division of the mantissa region $(1 \leq x < 2)$. The number of the required terms of Taylor-series expansion for a specified accuracy is reduced as the number of the division increases, and the addressing to the look-up table (LUT) is simple. (ii) The second is non-uniform division (exploration of all spaces and optimal) of the mantissa region. The region is optimally divided and the number of the required terms can be reduced, though the LUT addressing becomes complicated. (iii) The third is the mantissa region conversion. The mantissa is multiplied by 2, 4 or 8 and the region of $(2 \leq \mathrm{x} < 4), (4 \leq \mathrm{x} < 8)$ or $(8 \leq \mathrm{x} < 16)$ is considered and there Taylor-series expansion is applied. There the slope of $\mathrm{f(x)} = \log_2 \mathrm{x}$ with respect to x is gentle compared to in the region of $(1 \leq \mathrm{x} < 2)$, so that the required number of the terms can be reduced. Their hardware implementation is considered;

numbers of multiplications/additions/subtractions and the LUT contents, size and addressing. We show their design trade-off among accuracy and numbers of multiplications/additions /subtractions and LUT sizes. Also, an extreme case of the uniform division that the required number of the terms is 2 is shown; there the numbers of multiplications/additions/subtraction are small, whereas the LUT size is large. The designer can choose the optimal algorithm for his digital logarithmic calculation, and build its conceptual dedicated hardware architecture design with the contents described here.

Keywords: Floating-point, Logarithm, Digital arithmetic, Taylor-series expansion, Mantissa region division, Mantissa region conversion

## 7.2 Introduction

The data types of processors are divided into two families: floating-point and integer. In some cases, the floating-point number is preferred thanks to its wide dynamic range, though it requires complicated hardware or takes long calculation time. For instance, one floating-point addition involves exponent processing, shift and leading-zero count; it typically requires 3 to 6 cycles [1]. Even so, the floating-point arithmetic is widely used in many applications such as digital signal processing, image processing, biomedical applications and scientific computing [2-8]. It can be implemented in software and hardware, e. g., general purpose DSP, FPGA and CMOS full custom VLSI [9-14]. There, the basic arithmetic operations, such as addition, subtraction, multiplication, division, square root, and exponential are key operations, and also the logarithmic arithmetic plays an

important role for simpler exponentiation, division and multiplication computation [15-17].

Logarithmic simplicity indicates that multiplication and division can be performed using base-2 binary logarithmic as introduced in [18]; its authors presented a method for finding the approximate value of the $\log_2$ of a binary number based for intervals between the powers of two. However, the straight-line-approximation method limits the accuracy. Usage of the redundant logarithmic arithmetic relies on the table lookup to make the arithmetic unit simpler than the equivalent floating-point unit, and its disadvantages are typical ones for the redundant number systems introduced in [19]. Reference [20] describes the designs of both non-iterative and iterative approximate logarithmic multipliers (ALMs), which are studied to further reduce power consumption and improve performance. Their implementation is still restricted by complexity of performing addition and subtraction functions as a result of using lookup tables. The authors of the reference [20] showed that a method is revealed to substantially reduce the sizes of these tables using second-order co-transformation procedure in [21]. The potential of the power consumption reduction in digital system using the logarithmic number system (LNS) is investigated in [22]. The comparison of experiments showed that LNS reduces the assertion probability by more than 50 %. Finally, the authors of the reference [21] presented a method called Floor Shift for fast calculation of $\log_2$, and then this algorithm is combined with Taylor-series to improve the accuracy of the output with two examples [23]. Reference [24] describes a generalized hyperbolic COordinate Rotation

Digital Computer (GH CORDIC) to directly compute logarithms and exponentials with an arbitrarily fixed base, optimized the shortcomings of only using CORDIC. The uniform domain division of this method is well applied in [25,26], but there is no description of the unequal division. For calculating floating-point methods, there are also Monte Carlo, Look-up tables, CORDIC [27-30], and so on.

Here, we use the convergence of Taylor-series expansion with the mantissa region division and conversion for the floating-point logarithmic calculation. Three techniques are proposed; (i) The first one is uniform division of the mantissa region $(1 \leq x < 2)$. The number of the required terms of Taylor-series expansion for a specified accuracy is reduced as the number of the division increases, and the addressing to the look-up table (LUT) is simple. (ii) The second is non-uniform division of the mantissa region. The region is optimally divided and the number of the required terms can be reduced, though the LUT addressing becomes complicated. (iii) The third is the mantissa region conversion. The mantissa is multiplied by 2, 4 or 8 and the region of $(2 \leq x < 4), (4 \leq \mathrm{x} < 8)$ or $(8 \leq \mathrm{x} < 16)$ is considered and there Taylor-series expansion is applied. There the slope of $f(x) = \log_2 x$ with respect to $x$ is gentle compared to in the region of $(1 \leq \ x < 2)$ so that the required number of the terms can be reduced. Their hardware implementation is considered; the numbers of multiplications/additions/subtractions and the LUT contents, size and addressing. We show their design trade-off among accuracy and numbers of multiplications/ additions /subtractions and LUT sizes.

Also, an extreme case of the uniform division that the required number of the terms is 2 is shown; there the numbers of multiplications/additions/ subtraction are small, hence the logarithm calculation can be done fast, though the LUT size is large. This can be interpreted as an LUT based logarithmic calculation method with the interpolation calculation.

The suitable implementation would depend on the available hardware environment. The designer can choose the optimal algorithm for his digital logarithmic calculation, and build its conceptual dedicated hardware architecture design with the contents described here, which cover wide range of the logarithmic arithmetic algorithms and can compete other methods.

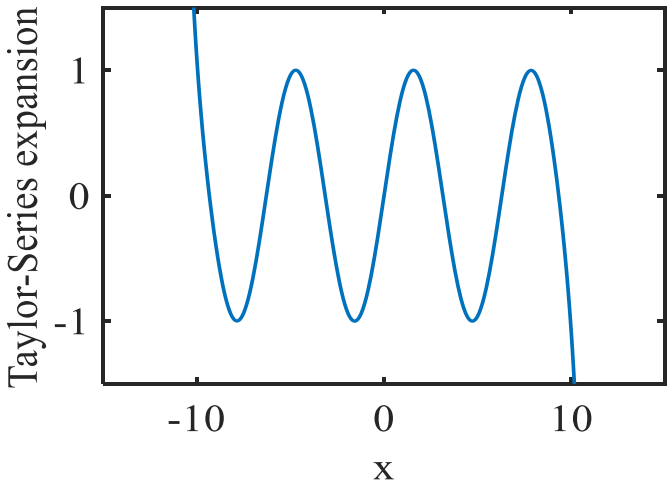
This chapter is organized as follows: Section 7.3 explains Taylor-series expansion, and Section 7.4 shows floating-point logarithmic arithmetic. Section 7.5 explains Taylor-series expansion method for floating-point logarithmic arithmetic. Sections 7.6 and 7.7 describe our uniform and non-uniform division methods for mantissa region division respectively. In Section 7.8, our mantissa region conversion method is investigated. Section 7.9 explains the mantissa region division for high-speed logarithmic calculation, and Section 7.10 shows verification of our methods with some examples. In Section 7.11, hardware implementation of our algorithms is considered and Section 7.12 provides summary.

## 7.3 Taylor-series Expansion

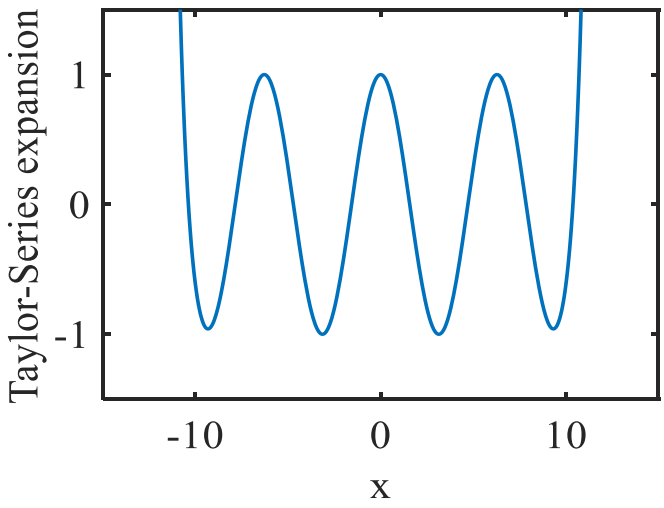
Consider an infinitely differentiable function $f(x)$, and its Taylor-series expansion at $x = a$ is given by :

$$f(x) = f(\mathrm{a}) + f'(a)(x-a) + \frac{(\mathrm{f})''(a)}{2!}(x-a)^2 + \cdots + \frac{(\mathrm{f})^n(a)}{\mathrm{n!}}(x-a)^{\mathrm{n}} + \cdots \tag{7-1}$$

Notice that many functions used in engineering design have relatively wide convergence radius; as *n* increases, the Taylor-series expansion reaches $f(x)$ for wide range of *x*. For examples, Taylor-series expansions at $a = 0$ of sine and cosine functions converge to sine and cosine for $-\infty < x < \infty$, respectively, as *n* increases (Fig. 7.1).




(a) sin(x)




(b) cos(x)

Fig. 7.1 Waveforms of Taylor-series expansion of $\sin(x)$ and $\cos(x)$ at $a = 0$ up-to 25 terms.

However, as far as we know, there are very few algorithms in the engineering field that positively utilize the fact that Taylor-series expansion of some $f(x)$ converges to $f(x)$ over a wide range of $x$. Then we investigate here a digital arithmetic algorithm for the logarithmic function $f(x)$ by its approximating with Taylor-series expansion.

## 7.4 Floating-point Logarithmic Arithmetic

### 7.4.1 Representation of Floating-point Number

The IEEE-754 format is a floating-point number which consists of three parts, namely the sign bit *S*, the exponent *E*, and the mantissa *M* as shown in Fig. 7.2.

| 31 | 30 | 23 | 22 |
|---|---|---|---|
| ***S*** | ***E (Exponent)*** | ***M (Mantissa)*** | |

Fig. 7.2 IEEE-754 floating-point format (single precision case).

This standard allows the user to work not only with 32-bit single precision, but also 16-bit half precision, 64-bit double precision in [31-33]; its precision can be selected by the designer.

Here, we consider only a positive number $X$; the sign bit always indicates "plus". Let $X, M$ and $E$ denote the floating-point representation, the mantissa field, and the exponent field in binary representation, respectively. So, the binary floating number $X$ can be expressed as:

$$X = M \times 2^E \tag{7-2}$$

Here, the mantissa is represented by $M = 1.\alpha\beta\gamma\cdots$ ($\alpha, \beta, \gamma\cdots$ is 0 or 1). Notice that $1 \leq M < 2$, and for example, $M = 1.011001$ (binary) = $1.390625$ (decimal).

The logarithm function is one of the most useful elementary functions. It is continuous and monotonically increasing. Now let us consider the floating-point algorithm, which calculates the logarithmic of the binary floating-point representation, expressed as $L$:

$$L = \log_2 x = \log_2 M + E \tag{7-3}$$

### 7.4.2 Mantissa and Exponent Parts for Base-2 Logarithm

Let $log_2 M + E$ be normalized to represent that its floating-point expression can be obtained as follows:

$$log_2 M + E = M_L \times 2^{E_L} \tag{7-4}$$

Here, $M_L$ and $E_L$ are the mantissa part and the exponent part, respectively.

## 7.5 Taylor-series Expansion Method for Floating-point Base-2 Logarithmic Arithmetic

### 7.5.1 Mantissa Part for Logarithm Calculation

Here, we concentrate on the mantissa field for the accurate base-2 logarithm calculation to satisfy a specified accuracy, based on the Taylor-series expansion with the mantissa region division.

Let us consider the calculation of $\log_2 M\ (1 \le M < 2)$ using Taylor-series expansion of the logarithm, with $x = a\ \ (1 \le a < 2)$ satisfying the specified accuracy. Taylor-series expansion of $f(x) = \log_2 x$ for $x = a$ is given as follows:

$$\begin{aligned} f(x) &= \frac{1}{\ln(2)}\left\{\ln(a) + \frac{p}{a} - \frac{p^2}{2 \times a^2} + \frac{p^3}{3 \times a^3} - \frac{p^4}{4 \times a^4} + \cdots\right\} \\ &= \frac{1}{\ln(2)}\left\{\ln(a) + \frac{p}{a}\left(1 - \frac{p}{a}\left(\frac{1}{2} + \frac{p}{a}\left(\frac{1}{3} - \frac{p}{a}\left(\frac{1}{4} + \cdots\right)\right)\right)\right)\right\} \end{aligned} \tag{7-5}$$

Here, $p = x - a$. Eq. (7-5) is to reduce the number of multiplications in Taylor-series expansion.

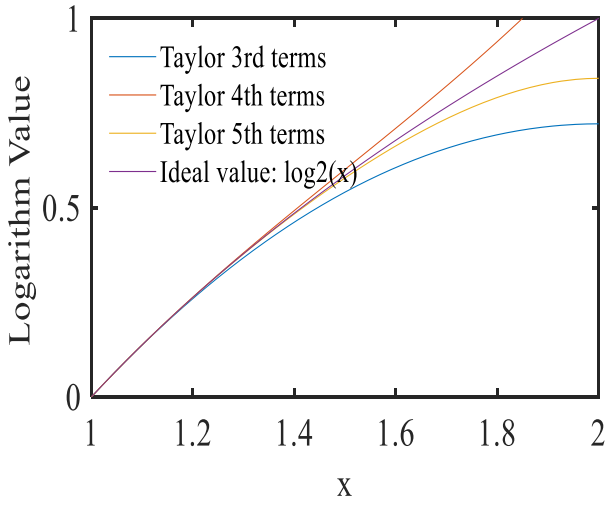
Fig. 7.3 (a) shows the graph of $f(x) = \log_2 x$ using Taylor-series expansion centered at $a = 1$ when taking 3rd, 4th and 5th terms produces the approximated value of the logarithm for $1 \leq x < 2$; notice that the output value accuracy increases as the number of the terms increases.

The approximate error in this case ($a = 1$) is shown in Fig. 7.3(b). Our definition of the approximate error is given by the following:

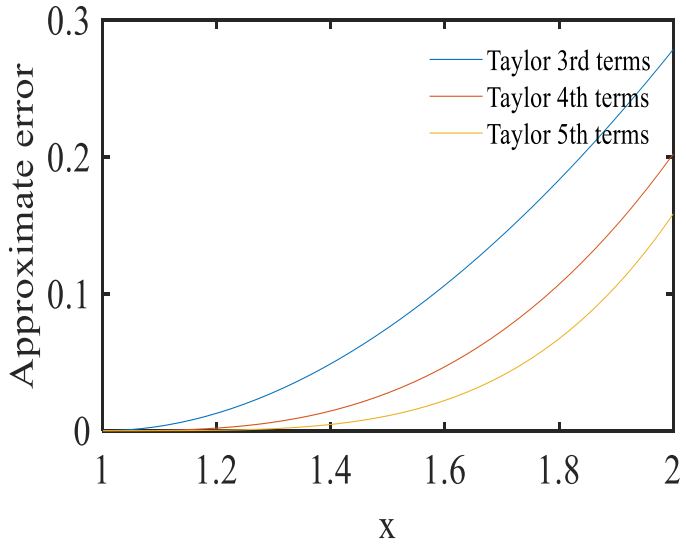
$$\text{Approximate Error} = \left| \frac{f(x) - t_n(x)}{f(x)} \right| \tag{7-6}$$

Here, $f(x)$ is the original function value and $t_n(x)$ is Taylor-series expansion value with *n* terms.

Similarly, for $f(x) = \log_2(x)$ using Taylor-series expansion centered at $a = 1.5$, we obtain Fig. 7.4(a) and 7.4(b).

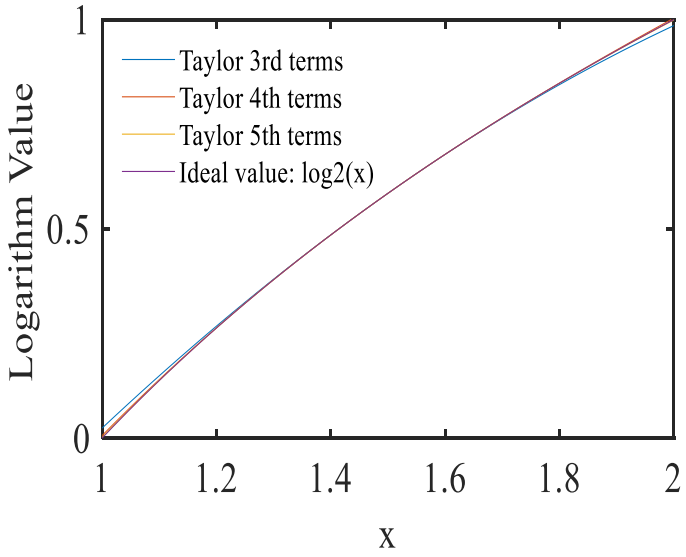



(a) Ideal and Taylor-series expansion values for $f(x) = \log_2(x)$.

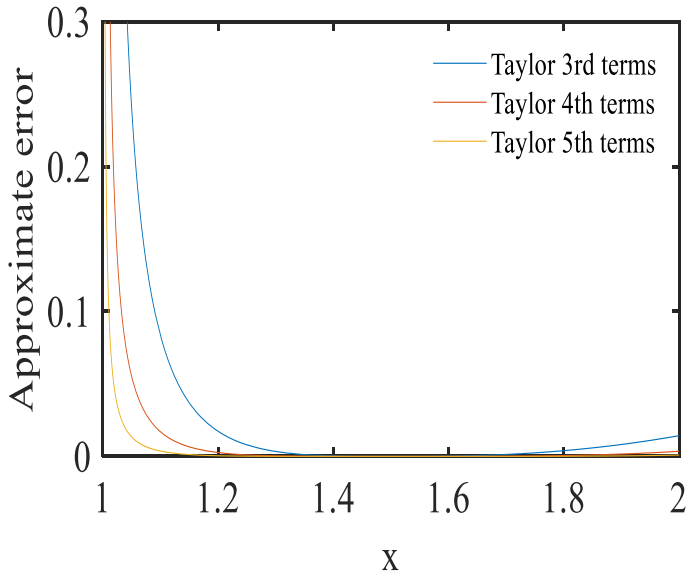



(b) Approximate error.

Fig. 7.3 Taylor-series expansion centered at $a = 1$.




(a) Ideal and Taylor-series expansion values for $f(x) = \log_2(x)$




(b) Approximate error

Fig. 7.4 Taylor-series expansion centered at $a = 1.5$.

We see by comparison of Figs. 7.3 and 7.4 that accurate results can be obtained near the center value of $a$, so that we use the center point value in the specified range, such as the center point of 1.5 in the range of $(1 \leq x < 2)$ as the center value $a$ of the Taylor-series expansion for accuracy. Based on this observation, we propose a division technique of the mantissa region $(1 \leq x < 2)$ for Taylor-series expansion usage. Notice that there is a special case or a kind of singularity at $f(1) = \log_2 1 = 0$, where the number of Taylor-series expansion terms is very large for good accuracy. In Sections 7.6 and 7.7, we analyze only the region of $1 < x < 2$, excluding $x$=1, but Section 7.8 shows its countermeasure.

### 7.5.2 Numerical Simulation Results of Logarithmic Function Taylor-series Expansion

In this section, we show our uniform division method for the region of $1 < x < 2$ based on Taylor-series expansion to calculate the mantissa of the logarithm.

We have performed simulation to obtain the number of terms $n$ expanded by Taylor-series of $f(x) = \log_2(x)$ for specified accuracy $p$ and given region, which satisfies the following:

$$max\left|\frac{f(x) - t_n(x)}{f(x)}\right| \leq p \tag{7-7}$$

Fig. 7.5 shows the convergence range in several cases using Taylor-series expansion $f(x) = \log_2(x)$. We see through comparative analysis that as the center point becomes larger, the convergence range becomes larger.

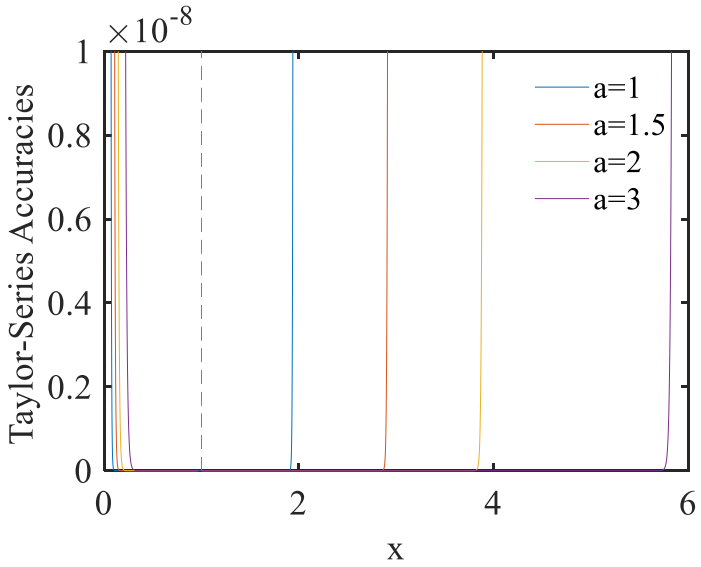



Fig. 7.5 Taylor-series expansion numerical calculations with different center values of $a$ (=1, 1.5, 2, 3) at $f(x) = \log_2 x$. Number of terms is 200.

## 7.6 Uniform Division of Mantissa Region

### 7.6.1 One Region of $1 < x < 2$

Table 7.1 shows the required number of the terms *n* for Taylor-series expansion to meet the specified accuracy, obtained by numerical simulation.

A-1-a) Taylor-series expansion of $f(x) = log_2 x$ at $a$ = 1.5 (1<$x$<2)

When the given accuracy of $p$ in Eq. (7-7) is $1/2^{16}$, we obtain $n$ = 18 for all $x$ (1< $x$ <2) from numerical simulations.

Table 7.1 Number of Taylor-series expansion terms to meet specified accuracy for one region of $1 < x < 2$.

| Accuracy / Taylor-series expansion | $\frac{1}{2^8}$ | $\frac{1}{2^{10}}$ | $\frac{1}{2^{16}}$ | $\frac{1}{2^{22}}$ | $\frac{1}{2^{24}}$ |
|---|---|---|---|---|---|
| A-1-a) | 13 | 15 | 18 | 22 | 23 |

### 7.6.2 Two Uniform Regions of $1 < x < 2$.

Fig. 7.6 explains the uniform division of mantissa region by 2.

We divide the region of $1 < x < 2$ by 2 evenly and choose the region, based on *M*. Then we perform the Taylor-series expansion at the point $a$ of the center for each region.

A-2-a) For $M$ = 1.0∗∗∗⋯ $(1 < M < 1.5)$, $a$=1.25.

A-2-b) For $M$ = 1.1∗∗∗⋯ $(1.5 \leq M < 2)$, $a$=1.75.

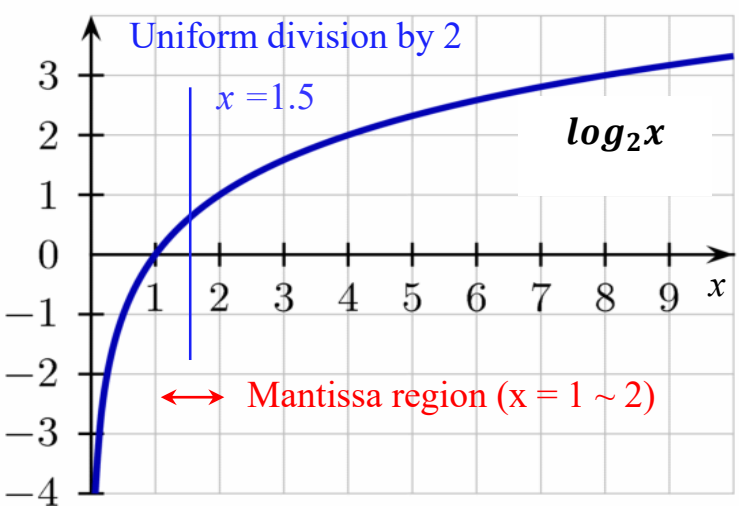



Fig. 7.6 Explanation of uniform division of mantissa region by 2.

Table 7.2 shows the numerical simulation results.

Table 7.2 Number of Taylor-series expansion terms to meet specified accuracy when the region of $1 < x < 2$ is divided by 2.

| Accuracy / Taylor-series expansion | $\frac{1}{2^8}$ | $\frac{1}{2^{10}}$ | $\frac{1}{2^{16}}$ | $\frac{1}{2^{22}}$ | $\frac{1}{2^{24}}$ |
|---|---|---|---|---|---|
| A-2-a) | 10 | 11 | 13 | 16 | 16 |
| A-2-b) | 4 | 5 | 7 | 9 | 10 |

### 7.6.3 Four Uniform Regions of $1 < x < 2$.

We divide the region of $1 < x < 2$ by 4 uniformly and choose the region, based on $M$. Then we perform the Taylor-series expansion at the point $a$ of the center of each divided region.

A-3-a) For $M$ = 1.00∗∗∗∗ ⋯ $(1 < M < 1.25)$, $a$=1.125.

A-3-b) For $M$ = 1.01∗∗∗∗ ⋯ $(1.25 \leq M < 1.5)$, $a$=1.375.

A-3-c) For $M$ = 1.10∗∗∗∗ ⋯ $(1.5 \leq M < 1.75)$, $a$=1.625.

A-3-d) For $M$ = 1.11∗∗∗∗ ⋯ $(1.75 \leq M < 2)$, $a$=1.875.

Table 7.3 shows the numerical simulation results.

Table 7.3 Number of Taylor-series expansion terms to meet specified accuracy when the region of $1 < x < 2$ is divided by 4.

| Taylor-series expansion \ Accuracy | $\frac{1}{2^8}$ | $\frac{1}{2^{10}}$ | $\frac{1}{2^{16}}$ | $\frac{1}{2^{22}}$ | $\frac{1}{2^{24}}$ |
|---|---|---|---|---|---|
| A-3-a) | 8 | 8 | 10 | 12 | 12 |
| A-3-b) | 4 | 5 | 6 | 8 | 8 |
| A-3-c) | 4 | 4 | 6 | 7 | 8 |
| A-3-d) | 4 | 4 | 5 | 7 | 7 |

## 7.7 Non-uniform Division for Mantissa Region

### 7.7.1 Basic Idea

In this section, we show our optimal domain division method for the region of $1 < x < 2$ based on Taylor-series expansion to calculate the mantissa of the logarithm. For example, Fig. 7.7 explains the optimal division of mantissa region by 2.

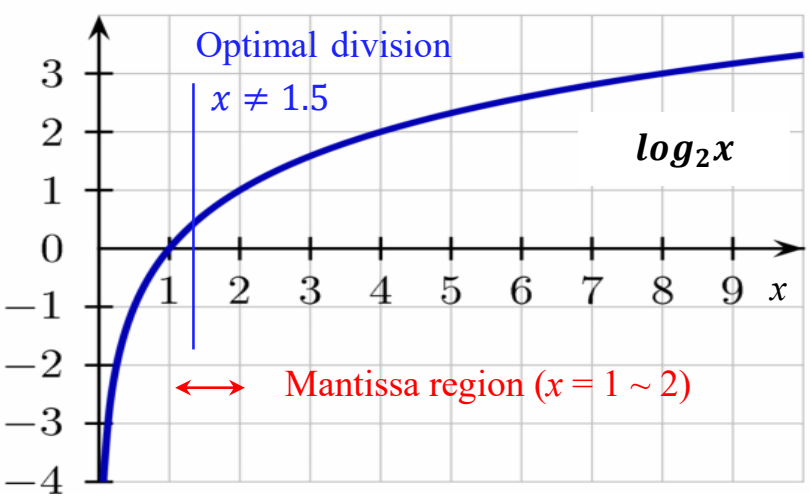



Fig. 7.7 Explanation of the optimal division of mantissa region by 2.

We see in Fig. 7.4(b) that when the region $1 < x < 2$ and $a = 1.5$, the error on the left is larger than that on the right. Based on this finding, we move the mantissa region division boundaries and their center points for the Taylor-series expansion to the left, so that errors in divided regions are balanced.

We start the two non-uniform region division using a bisection method for optimal partitioning, and we consider the region I: $1 < x < x_1$ and the region II: $x_1 \leq x < 2$. Fig. 7.8 shows the algorithm to obtain the boundary value of $x_1$ for the minimum number of the Taylor-series expansion terms for a specific accuracy. Also we obtain the minimum terms N1 and N2 so that $\mathrm{N1} = \mathrm{N2}$ or $|N1 - N2| = 1$. This is similar in four and eight region division cases.

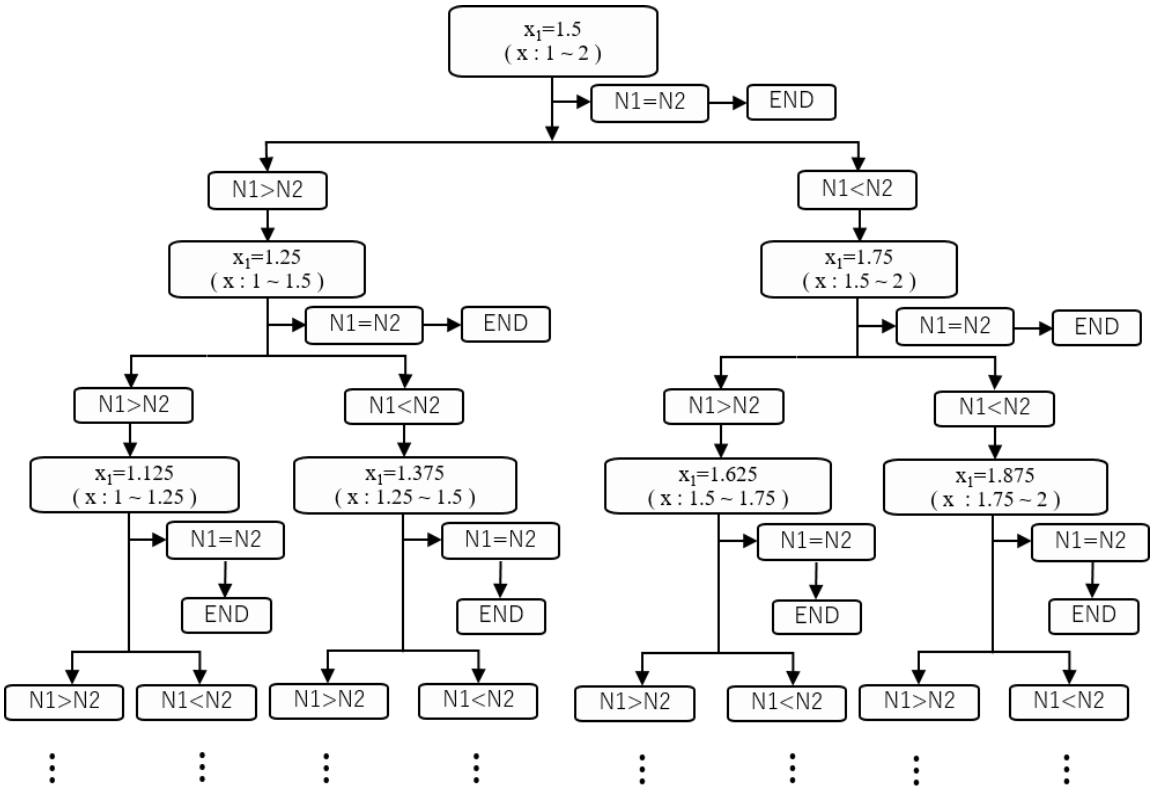



Fig. 7.8 Exploration of all mantissa spaces and optimal flowchart

### 7.7.2 One Region of $1 < x < 2$.

Table 7.4 shows the required number of the terms *n* for Taylor-series expansion to meet the specified accuracy, obtained by numerical simulation.

B-1-a) Taylor-series expansion of $f(x) = \log_2 x$ at $a$ =1.25 $(1 < x < 2)$.

Table 7.4 Number of Taylor-series expansion terms to meet specified accuracy for one region of $1 < x < 2$.

| Accuracy / Taylor-series expansion | $\frac{1}{2^8}$ | $\frac{1}{2^{10}}$ | $\frac{1}{2^{16}}$ | $\frac{1}{2^{22}}$ | $\frac{1}{2^{24}}$ |
|---|---|---|---|---|---|
| B-1 | 7 | 9 | 15 | 20 | 22 |

### 7.7.3 Two Non-uniform Regions of $1 < x < 2$.

We divide the region of $1 < x < 2$ by 2 optimally and choose the region, based on *M*, and perform the Taylor-series expansion at the point *a* of the center for each region.

B-2-a) For $M = 1.{*}{*}{*}{*}\cdots$ $(1 < M < 1.25)$, $a$=1.125.

B-2-b) For $M = 1.{*}{*}{*}{*}\cdots$ $(1.25 \leq M < 2)$, $a$=1.625.

Table 7.5 shows the numerical simulation results. Compared to the uniform division A-2-a) case in Table 2, the required number of terms is reduced substantially; for example, in 24-bit accuracy case, it is reduced from 16 to 12.

Table 7.5 Number of Taylor-series expansion terms to meet specified accuracy when the region of $1 < x < 2$ is divided by 2.

| Taylor-series expansion \ Accuracy | $\frac{1}{2^8}$ | $\frac{1}{2^{10}}$ | $\frac{1}{2^{16}}$ | $\frac{1}{2^{22}}$ | $\frac{1}{2^{24}}$ |
|---|---|---|---|---|---|
| B-2 | 7 | 7 | 9 | 11 | 12 |

### 7.7.4 Four Non-uniform Regions of $1 < x < 2$.

Divide the region of $1 < x < 2$ by 4 optimally and choose the region, based on *M*. Then perform the Taylor-series expansion at the point *a* of the center of each divided region.

B-3-a) For $M = 1.{*}{*}{*}{*}\cdots$ $(1 < M < 1.03125)$, $a$=1.015625.

B-3-b) For $M$=1.$\ast\ast\ast\ast\cdots$ $(1.03125 \leq M < 1.25)$, $a$=1.140625.

B-3-c) For $M$ = 1.$\ast\ast\ast\ast\cdots$ $(1.25 \leq M < 1.5625)$, $a$=1.40625.

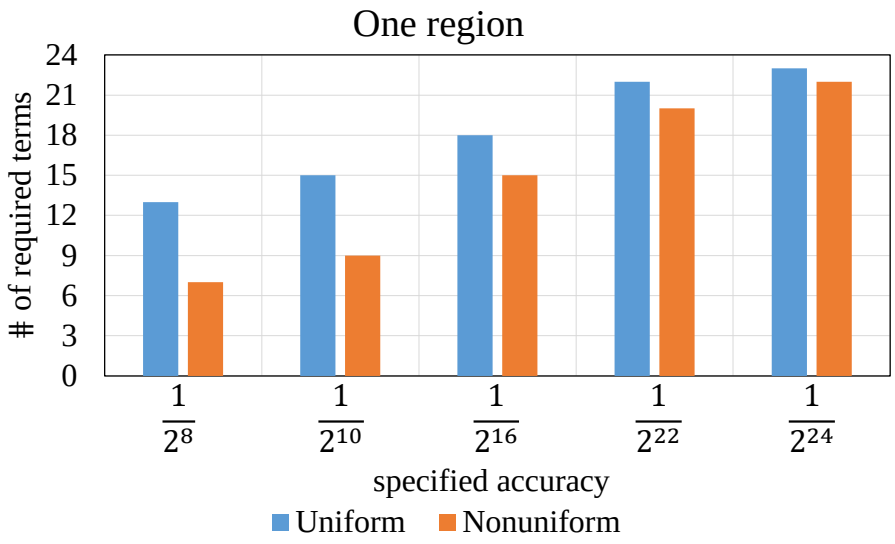
B-3-d) For $M$ = 1.$\ast\ast\ast\ast\cdots$ $(1.5625 \leq M < 2)$, $a$=1.78125.

Table 7.6 shows the numerical simulation results. Compared to the uniform division A-3-a) case in Table 7.3, the required number of terms is reduced substantially; for example, in 24-bit accuracy case, it is reduced from 12 to 8.

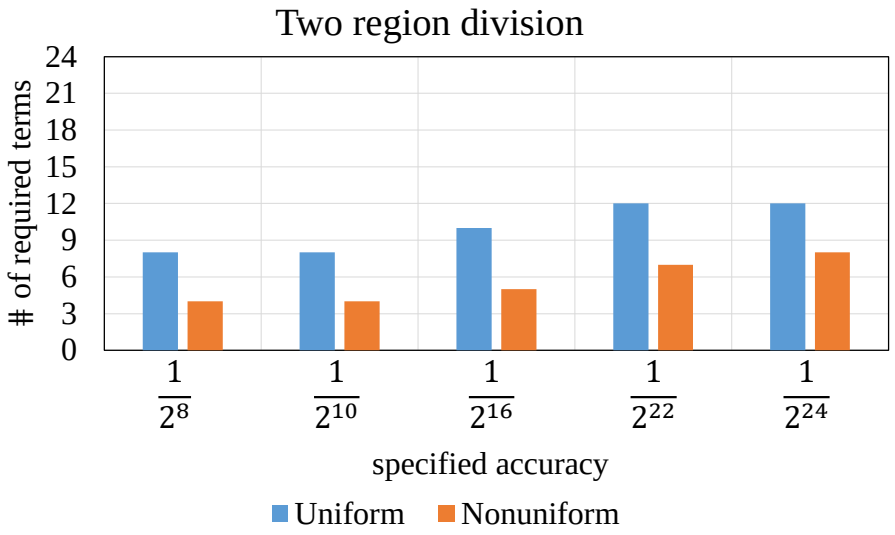
Table 7.6 Number of Taylor-series expansion terms to meet specified accuracy when the region of $1 < x < 2$ is divided by 4.

| Accuracy / Taylor-series expansion | $\frac{1}{2^8}$ | $\frac{1}{2^{10}}$ | $\frac{1}{2^{16}}$ | $\frac{1}{2^{22}}$ | $\frac{1}{2^{24}}$ |
|---|---|---|---|---|---|
| B-3 | 4 | 4 | 5 | 7 | 8 |

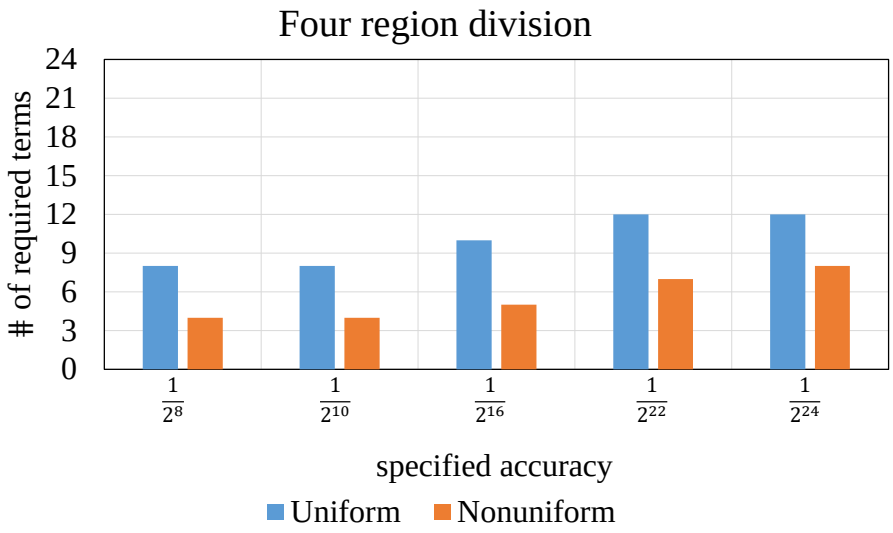
Fig. 7.9 shows the required number of the Taylor-series expansion obtained by several region divisions with uniform and non-uniform methods under the specified accuracy. It is the maximum number of expansion terms among the divided regions for the accuracy. For example, Table 7.4 shows that the maximum number of expansion terms for 22-bit accuracy is 12. We see from Fig. 7.9 that the non-uniform division method is effective for the reduction of the number.




(a) One region.




(b) Region division by 2




(c) Region division by 4.

Fig. 7.9 Taylor-series expansion obtained by several region divisions with uniform and non-uniform methods under the specified accuracy.

## 7.8 Mantissa Region Conversion

### 7.8.1 Basic Idea

We consider here that the mantissa is multiplied by 2, 4 or 8 and the region of $(2 \leq x < 4)$, $(4 \leq \mathrm{x} < 8)$ or $(8 \leq \mathrm{x} < 16)$ is considered and there Taylor-series expansion is applied. There the slope of $f(x) = \log_2 x$ with respect to $x$ is gentle compared to in the region of $(1 \leq x < 2)$, so that the required number of the terms can be reduced.

Also we see in Fig. 7.4 that the Taylor-series expansion requires many terms, to obtain $f(1) = \log_2 1 = 0$ with the Taylor-series expansion with the center point of $a$=1.5 for $1 \leq x < 2$. If we consider the Taylor-series

expansion region of $(2 \le x < 4)$, $(4 \le x < 8)$ or $(8 \le x < 16)$, this problem can be avoided.

Our proposed region conversion method is to move the mantissa region from $1 \le x < 2$ to $b \le x_m < 2b$ ($b = 2^m$ and $m = 1, 2, 3 \cdots$), and another expression of $f(x) = \log_2 x$ is obtained as follows:

$$\begin{aligned} f(x) = \log_2 x &= \log_2 \left(\frac{bx}{b}\right) \\ &= \log_2 (bx) - \log_2 b \\ &= \log_2 x_m - m \end{aligned} \tag{7-8}$$

Here, $x_m = bx$.

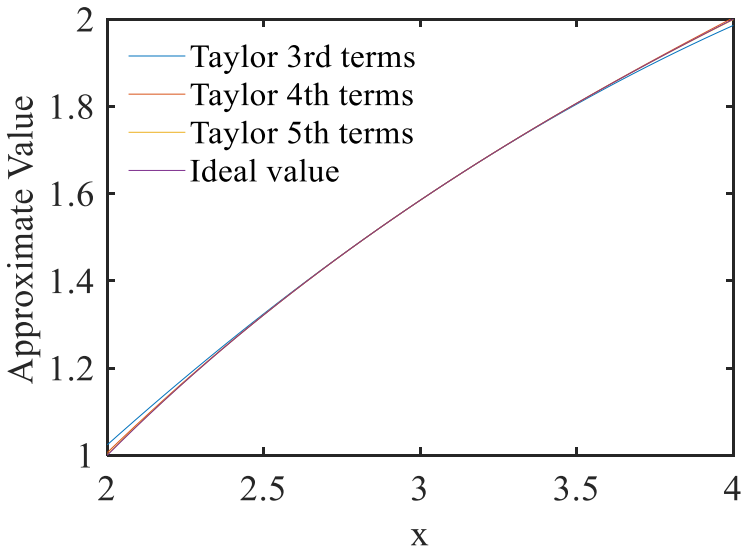
We consider the Taylor-series expansion of $f_m(x_m) = \log_2 x_m$ at the center value $a$ $(b \le a < 2b)$ to solve $f(x) = \log_2 x$. We obtain the following:

$$\begin{aligned} f_m(x_m) &= \frac{1}{\ln(2)} \left\{ \ln(\mathrm{a}) + \frac{p_m}{\mathrm{a}} - \frac{{p_m}^2}{2 \times \mathrm{a}^2} + \frac{{p_m}^3}{3 \times \mathrm{a}^3} - \frac{{p_m}^4}{4 \times \mathrm{a}^4} + \cdots \right\} \\ &= \frac{1}{\ln(2)} \left\{ \ln(\mathrm{a}) + \frac{p_m}{a} \left(1 - \frac{p_m}{a} \left(\frac{1}{2} + \frac{p_m}{a} \left(\frac{1}{3} - \frac{p_m}{a} \left(\frac{1}{4} + \cdots\right)\right)\right)\right) \right\} \end{aligned} \tag{7-9}$$

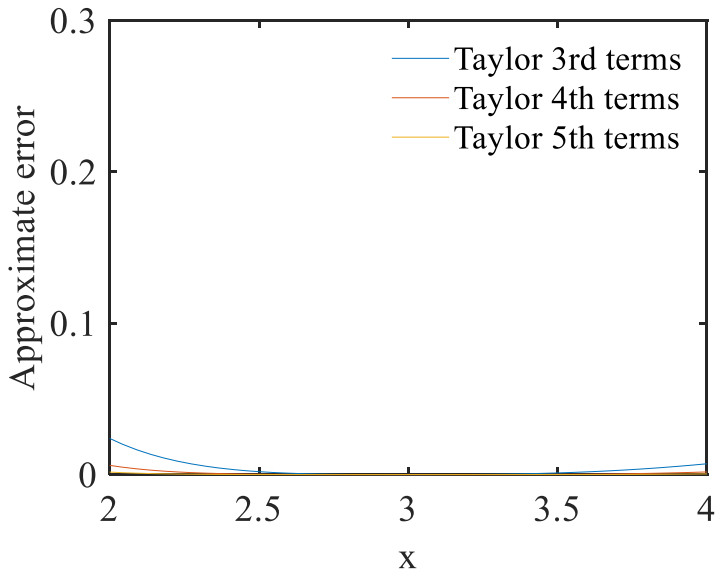
$$\begin{aligned} f(x) &= \log_2 x_m - m \\ &= f_m(x_m) - m \end{aligned}$$

Here, $p_m = x_m - a$. Let $m$ = 1, and we obtain $f(x) = f_m(x_m) - 1$. Now we analyzer Taylor-series expansion from $f_m(x_m)$ in the range of $2 \le x_m < 4$.

Fig. 7.10(a) shows the graph of $f_{\mathrm{m}}(x_m)$ using Taylor-series expansion centered at $a = 3$ when taking 3rd, 4th and 5th terms produces the approximated value of the logarithm for $2 \leq x_m < 4$ and its expansion approximate error is shown in Fig. 7.10(b). We see from comparison with Fig. 7.4 that the accuracy of the mantissa region conversion method is better. Then Fig. 7.11 explains the region conversions $(m = 1, 2)$.




(a) Ideal and Taylor-series expansion values for $f(x_m) = \log_2(x_m)$.




(b) Approximate error.

Fig. 7.10 Taylor-series expansion centered at $a = 3$.

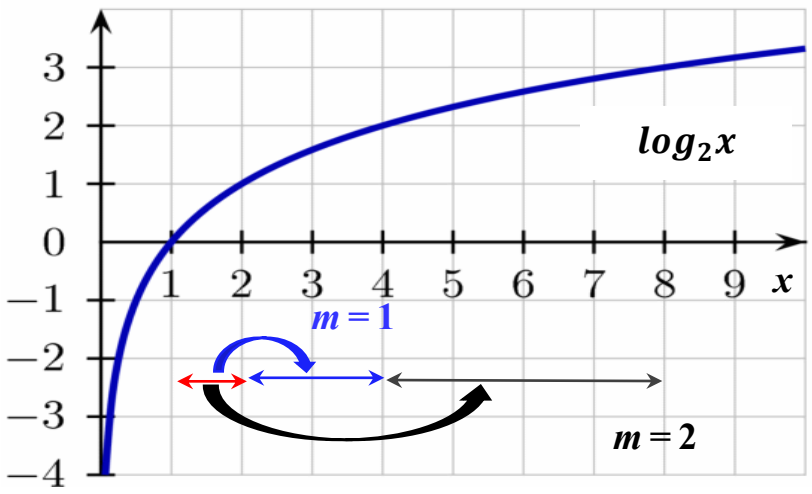



Fig. 7.11 Explanation of the region conversions $(m = 1, 2)$.

### 7.8.2 Mantissa Region Conversion to $2 \leq x_m < 4$ ($m$=1)

(1) One Region of $2 \leq x_m < 4$.

Table 7.7 shows the required number of the terms *n* for Taylor-series expansion to meet the specified accuracy, obtained by numerical simulation.

C-1-a) Taylor-series expansion of $f_m(x_m) = \log_2 x_m$ at $a$ = 3 $(2 \leq x_m < 4)$.

Table 7.7 Number of Taylor-series expansion terms to meet specified accuracy for one region of $2 \leq x_m < 4$.

| Accuracy / Taylor-series expansion | $\frac{1}{2^8}$ | $\frac{1}{2^{10}}$ | $\frac{1}{2^{16}}$ | $\frac{1}{2^{22}}$ | $\frac{1}{2^{24}}$ |
|---|---|---|---|---|---|
| C-1-a | 5 | 6 | 9 | 13 | 13 |

(2) Two Regions of $2 \leq x_m < 4$.

We divide the region of $2 \leq x_m < 4$ by 2 uniformly and choose the region, based on $M_1 (= 2 \times M)$ and perform the Taylor-series expansion at the point *a* of the center for each region.

C-2-a) For $M_1$ = 2.$\ast\ast\ast\ast \cdots$ $(2 \leq M_1 < 3)$, $a$=2.5.

C-2-b) For $M_1$= 3.$\ast\ast\ast\ast \cdots$ $(3 \leq M_1 < 4)$, $a$=3.5.

Table 7.8 shows the numerical simulation results.

(3) Four Regions of $2 \leq x_m < 4$.

Divide the region of $2 \leq x_m < 4$ by 4 uniformly and choose the region, based on $M_1$. Then perform the Taylor-series expansion at the point *a* of the center of each divided region.

C-3-a) For $M_1$ = 2.$\ast\ast\ast\ast \cdots$ $(2 < M_1 < 2.5)$, *a*=2.25.

C-3-b) For $M_1$ = 2.∗∗∗∗ ⋯ $(2.5 \leq M_1 < 3)$, $a$=2.75.

C-3-c) For $M_1$ = 3.∗∗∗∗ ⋯ $(3 \leq M_1 < 3.5)$, $a$=3.25.

C-3-d) For $M_1$ = 3.∗∗∗∗ ⋯ $(3.5 \leq M_1 < 4)$, $a$=3.75.

Table 7.9 shows the numerical simulation results.

Table 7.8 Number of Taylor-series expansion terms to meet specified accuracy when the region of $2 \leq x_m < 4$ is divided by 2.

| Accuracy / Taylor-series expansion | $\frac{1}{2^8}$ | $\frac{1}{2^{10}}$ | $\frac{1}{2^{16}}$ | $\frac{1}{2^{22}}$ | $\frac{1}{2^{24}}$ |
|---|---|---|---|---|---|
| C-2-a) | 4 | 4 | 7 | 9 | 10 |
| C-2-b) | 3 | 4 | 5 | 7 | 8 |

Table 7.9 Number of Taylor-series expansion terms to meet specified accuracy when the region of $2 \leq x_m < 4$ is divided by 4.

| Accuracy / Taylor-series expansion | $\frac{1}{2^8}$ | $\frac{1}{2^{10}}$ | $\frac{1}{2^{16}}$ | $\frac{1}{2^{22}}$ | $\frac{1}{2^{24}}$ |
|---|---|---|---|---|---|
| C-3-a) | 3 | 3 | 5 | 7 | 7 |
| C-3-b) | 3 | 3 | 5 | 6 | 7 |
| C-3-c) | 2 | 3 | 4 | 6 | 6 |
| C-3-d) | 2 | 3 | 4 | 5 | 6 |

### 7.8.3 Mantissa Region Conversion to $4 \leq x_m < 8$ ($m$ = 2)

Here we analyze Taylor-series expansion of $f_m(x_m)$ in the range of

$4 \le x_m < 8$, and we obtain $f(x) = f_m(x_m) - 2$.

(1) One Region of $4 \le x_m < 8$ ($m$=2)

For $4 \le x_m < 8$ in one region, the results are shown in Table 7.10.

D-1-a) Taylor-series expansion of $f_m(x) = \log_2 x_m$ at $a$ = 6 ($4 \le x_m < 8$).

Table 7.10 Number of Taylor-series expansion terms to meet specified accuracy for one region of $4 \le x_m < 8$.

| Taylor-series expansion \ Accuracy | $\frac{1}{2^8}$ | $\frac{1}{2^{10}}$ | $\frac{1}{2^{16}}$ | $\frac{1}{2^{22}}$ | $\frac{1}{2^{24}}$ |
|---|---|---|---|---|---|
| D-1-a | 4 | 5 | 9 | 12 | 13 |

(2) Two Uniform Regions of $4 \le x_m < 8$

We divide the region of $4 \le x_m < 8$ by 2 uniformly and choose the region, based on $M_2 (= 4 \times M)$, and perform the Taylor-series expansion at the point $a$ of the center for each region.

D-2-a) Taylor-series expansion of $f_m(x) = \log_2 x_m$ at $a$ = 5 ($4 \le x_m < 6$).

D-2-b) Taylor-series expansion of $f_m(x) = \log_2 x_m$ at $a$ = 7 ($6 \le x_m < 8$).

Table 7.11 shows the numerical simulation results.

Table 7.11 Number of Taylor-series expansion terms to meet specified accuracy when the region of $4 \leq x_m < 8$ is divided by 2.

| Taylor-series expansion \ Accuracy | $\frac{1}{2^8}$ | $\frac{1}{2^{10}}$ | $\frac{1}{2^{16}}$ | $\frac{1}{2^{22}}$ | $\frac{1}{2^{24}}$ |
|---|---|---|---|---|---|
| D-2-a) | 3 | 4 | 6 | 9 | 9 |
| D-2-b) | 3 | 3 | 5 | 7 | 8 |

(3) Four Uniform Regions of $4 \leq x_m < 8$.

Divide the region of $4 \leq x_m < 8$ by 4 and choose the region, based on $m_x$. Then perform the Taylor-series expansion at the point *a* of the center of each divided region.

D-3-a) Taylor-series expansion of $f_m(x) = \log_2 x_m$ at $a$ = 4.5 ($4 \leq x_m < 5$).

D-3-b) Taylor-series expansion of $f_m(x) = \log_2 x_m$ at $a$ = 5.5 ($5 \leq x_m < 6$).

D-3-c) Taylor-series expansion of $f_m(x) = \log_2 x_m$ at $a$ = 6.5 ($6 \leq x_m < 7$).

D-3-d) Taylor-series expansion of $f_m(x) = \log_2 x_m$ at $a$ = 7.5 ($7 \leq x_m < 8$).

Table 7.12 shows the numerical simulation results.

Table 7.12 Number of Taylor-series expansion terms to meet specified accuracy when the region of $4 \leq x_m < 8$ is divided by 4.

| Taylor-series expansion \ Accuracy | $\frac{1}{2^8}$ | $\frac{1}{2^{10}}$ | $\frac{1}{2^{16}}$ | $\frac{1}{2^{22}}$ | $\frac{1}{2^{24}}$ |
|---|---|---|---|---|---|
| D-3-a) | 3 | 3 | 5 | 7 | 7 |
| D-3-b) | 2 | 3 | 4 | 6 | 7 |
| D-3-c) | 2 | 3 | 4 | 6 | 6 |
| D-3-d) | 2 | 3 | 4 | 5 | 6 |

### 7.8.4 Mantissa Region Conversion to $8 \leq x_m < 16$ $(m = 3)$

Here we analyze Taylor-series expansion of $f_m(x_m)$ in the range of $8 \leq x_m < 16$, and we obtain $f(x) = f_m(x_m) - 3$.

(1) One Region of $8 \leq x_m < 16$

For $8 \leq x_m < 16$ in one region, the results are shown in Table 7.13.

F-1-a) Taylor-series expansion of $f_m(x) = \log_2 x_m$ at $a$ = 12 $(8 \leq x_m < 16)$.

Table 7.13 Number of Taylor-series expansion terms to meet specified accuracy for one region of $8 \leq x_m < 16$.

| Taylor-series expansion \ Accuracy | $\frac{1}{2^8}$ | $\frac{1}{2^{10}}$ | $\frac{1}{2^{16}}$ | $\frac{1}{2^{22}}$ | $\frac{1}{2^{24}}$ |
|---|---|---|---|---|---|
| F-1-a | 4 | 5 | 9 | 12 | 13 |

(2) Two Uniform Regions of $8 \le x_m < 16$

We divide the region of $8 \le x_m < 16$ by 2 uniformly and choose the region, based on $M_3 (= 8 \times M)$, and perform the Taylor-series expansion at the point *a* of the center for each region.

F-2-a) Taylor-series expansion of $f_m(x) = \log_2 x_m$ at $a$ = 10 $(8 \le x_m < 12)$.

F-2-b) Taylor-series expansion of $f_m(x) = \log_2 x_m$ at $a$ = 14 $(12 \le x_m < 16)$.

Table 7.14 shows the numerical simulation results.

Table 7.14 Number of Taylor-series expansion terms to meet specified accuracy when the region of $8 \le x_m < 16$ is divided by 2.

| Accuracy / Taylor-series expansion | $\frac{1}{2^8}$ | $\frac{1}{2^{10}}$ | $\frac{1}{2^{16}}$ | $\frac{1}{2^{22}}$ | $\frac{1}{2^{24}}$ |
|---|---|---|---|---|---|
| F-2-a) | 3 | 4 | 6 | 9 | 9 |
| F-2-b) | 3 | 3 | 5 | 7 | 8 |

(3) Four Uniform Regions of $8 \le x_m < 16$.

Divide the region of $8 \le x_m < 16$ by 4 and choose the region, based on $m_x$. Then perform the Taylor-series expansion at the point *a* of the center of each divided region.

F-3-a) Taylor-series expansion of $f_m(x) = \log_2 x_m$ at $a$ = 9 $(8 \le x_m < 10)$.

F-3-b) Taylor-series expansion of $f_m(x) = \log_2 x_m$ at $a$ = 11 $(10 \leq x_m < 12)$.

F-3-c) Taylor-series expansion of $f_m(x) = \log_2 x_m$ at $a$ = 13 $(12 \leq x_m < 14)$.

F-3-d) Taylor-series expansion of $f_m(x) = \log_2 x_m$ at $a$ = 15 $(14 \leq x_m < 16)$.

Table 7.15 shows the numerical simulation results.

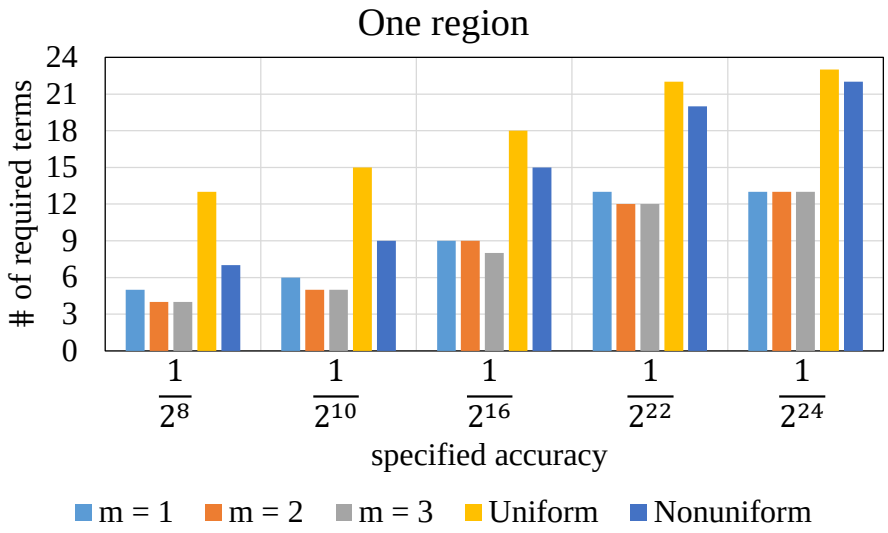
Table 7.15 Number of Taylor-series expansion terms to meet specified accuracy when the region of $8 \leq x_m < 16$ is divided by 4.

| Accuracy / Taylor-series expansion | $\frac{1}{2^8}$ | $\frac{1}{2^{10}}$ | $\frac{1}{2^{16}}$ | $\frac{1}{2^{22}}$ | $\frac{1}{2^{24}}$ |
|---|---|---|---|---|---|
| F-3-a) | 2 | 3 | 5 | 6 | 7 |
| F-3-b) | 2 | 3 | 4 | 6 | 6 |
| F-3-c) | 2 | 3 | 4 | 5 | 6 |
| F-3-d) | 2 | 2 | 4 | 5 | 6 |

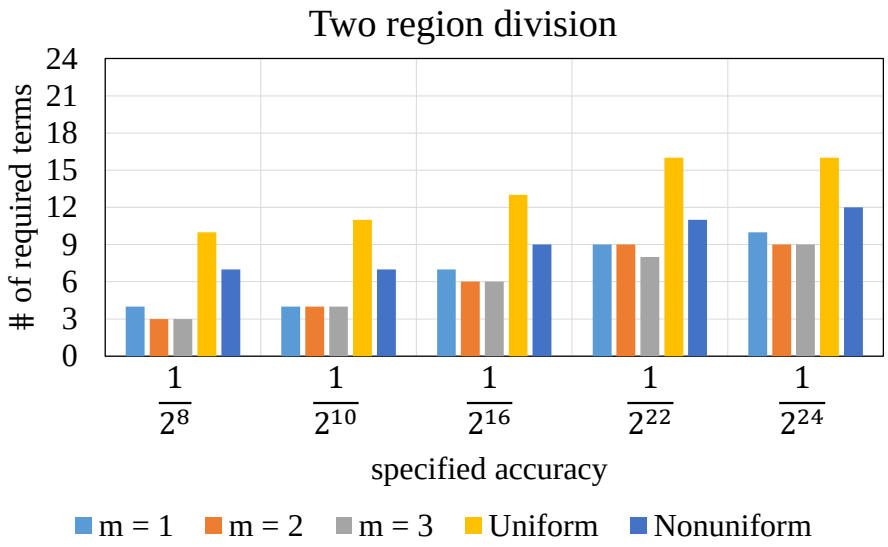
### 7.8.5 Comparison of Mantissa Region Conversions (*m*=1, 2, 3)

Fig. 7.12 shows the required number of the Taylor-series expansion obtained by several region divisions with mantissa conversions ($m$ = 1, 2, 3) using the uniform division, and non-mantissa conversion using the uniform and non-uniform divisions under the specified accuracy. We see that the mantissa conversion is effective for the required number of terms. Also, we

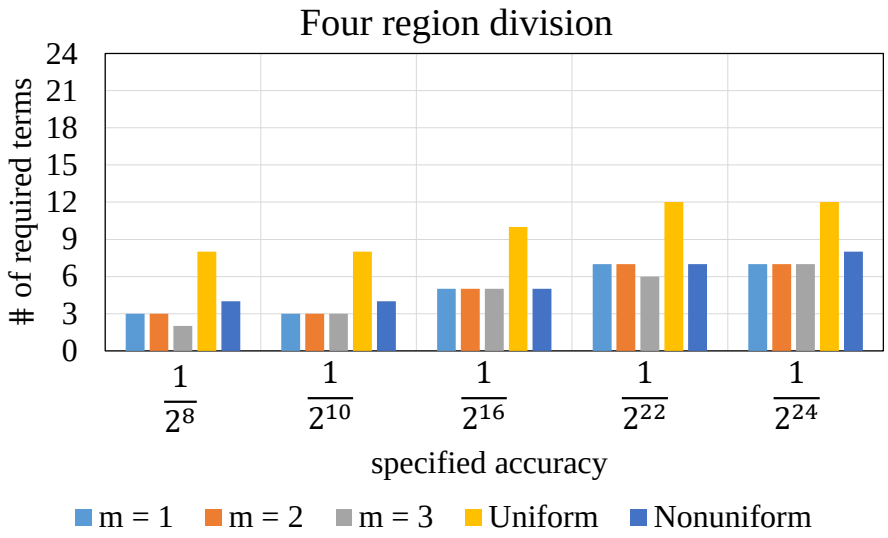
see that there is not substantial difference in $m = 1, 2, 3$ cases.




(a) One Region.




(b) Region division by 2.




(c) Region division by 4.

Fig. 7.12 Taylor-series expansion obtained by the uniform region division with the mantissa conversions ($m$=1, 2, 3) and the uniform and non-uniform region divisions without mantissa region conversion.

## 7.9 Mantissa Region Division for High-Speed Logarithmic Calculation

In this section, we investigate the logarithmic calculation to meet the specified accuracy with 2 terms of its Taylor-series expansion as follows:

$$f_2(x) = \frac{1}{\ln(2)} \times \left\{\ln(a) + \frac{x-a}{a}\right\} = \frac{1}{\ln(2)} \times \left\{\ln(a) + \frac{x}{a} - 1\right\} \quad (7\text{-}10)$$

Here, $1 / (\ln(2))$and $\ln(a)$ for each divided region are stored in LUT as discussed in Section 7.11. Take region conversion $m = 1$ as an example, and Table 7.16 shows the required number of mantissa region division for the specified accuracy. The calculation of 2-term Taylor-series expansion

requires only 2 multiplications and 2 additions/subtractions; hence its calculation can be done at high speed, though the LUT size becomes relatively large.

Table 7.16 Number of mantissa region division to meet specified accuracy with 2 terms of Taylor-series expansion.

| Accuracy | $\frac{1}{2^8}$ | $\frac{1}{2^{10}}$ | $\frac{1}{2^{16}}$ | $\frac{1}{2^{22}}$ | $\frac{1}{2^{24}}$ |
|---|---|---|---|---|---|
| Number of Mantissa Region Division | 8 | 16 | 126 | 1024 | 2048 |

## 7.10 Verification with Some Examples

In this section, we compare the direct calculation result and the Taylor-series expansion simulation result of the logarithmic calculation with some examples.

For example, for decimal number $L = \log_2 1.71875 + 8$, through calculation we can obtain the normal floating-point representation as follows:

$$L = 1.00001100100000000001110011_{(2)} \times 2^3$$

Here, we obtain the floating-point mantissa part of $1.00001100100000000001110011_{(2)}$ and the exponent part of 3.

We use the Taylor-series expansion method with the mantissa region division to calculate $\log_2 M$. For example, in the case of 8-bit accuracy and $M = 1.71875$ using 4 divided regions, the corresponding case of $M$ is

shown in A-3-c, and the Taylor-series with 4 terms at $a$ =1.625 is expanded as follows:

$$t(x) = \frac{1}{\ln(2)}\left\{\ln\left(\frac{13}{8}\right) + \frac{\left(x - \frac{13}{8}\right)}{\frac{13}{8}} - \frac{\left(x - \frac{13}{8}\right)^2}{2 \times \left(\frac{13}{8}\right)^2} + \frac{\left(x - \frac{13}{8}\right)^3}{3 \times \left(\frac{13}{8}\right)^3}\right\} \tag{7-11}$$

For $x$=1.71875, the following is obtained from Eq. (11):

$$t(1.71875) = 1.00001100100000000111011\mathbf{1}1_{(2)} \times 2^3$$

We see by their comparison that their mantissa parts $1.00001100100000000111011\mathbf{1}1_{(2)}$, and exponent parts $2^3$ of the direct and Taylor-series expansion calculation are the same. The error from the ideal value of $\log_2 1.71875 + 8$ is $4.888495\cdots \times 10^{-6}$ (which is less than $1/2^8$) using Eq. (7.7).

Therefore, we see that the mantissa part and the exponent part of $L$ are $1.00001100100000000111011\mathbf{1}1_{(2)}$ and 3, respectively.

## 7.11 Hardware Implementation Consideration

Let us consider the hardware implementation complexity using our algorithm to perform $f(x) = \log_2 x$ calculation in different cases.

Now let us consider the required numbers of multiplications

/additions/subtractions for Taylor-series expansion to calculate $L = \log_2 x = \log_2 M + E$. For example, in case of 5-term Taylor-series expansion $f_5(\mathrm{x})$ for $f(x) = \log_2 x$ at $x = a$, we have the following:

$$f_5(x) = \frac{1}{\ln(2)}\left\{ln(a) + \frac{p}{a} - \frac{p^2}{2 \times a^2} + \frac{p^3}{3 \times a^3} - \frac{p^4}{4 \times a^4}\right\} \quad (7\text{-}12)$$

$$= \alpha_0 + \alpha_1 \times p - \alpha_2 \times p^2 + \alpha_3 \times p^3 - \alpha_4 \times p^4$$

Here, $\alpha_0 = \frac{\ln(a)}{\ln(2)}$, $\alpha_1 = \frac{1}{a \times \ln(2)}$, $\alpha_2 = \frac{1}{2 \times a^2 \times \ln(2)}$, $\alpha_3 = \frac{1}{3 \times a^3 \times \ln(2)}$, $\alpha_4 = \frac{1}{4 \times a^4 \times \ln(2)}$.

Also $a$ is a constant and $x$ is a variable. $\alpha_0, \alpha_1, \alpha_2, \alpha_3, \alpha_4, \ldots$ are calculated in advance and stored in LUT memory; they are read at calculation time. Then we calculate $p = y = x - a$, $z = y^2$, and we have the following:

$$f_5(x) = a_0 + a_1 y - z(a_2 + a_3 y - a_4 z) \quad (7\text{-}13)$$

We see that $f_5$ can be obtained with 5 multiplications and 5 additions/ subtractions. Table 7.17 shows the required numbers of multiplications and additions/subtractions for the number of Taylor-series terms for $f(x) = \log_2 x$.

We see from Tables 7.6 and 7.17 that by dividing the region by 4, the logarithm of the mantissa can be calculated with 22-bit accuracy by 7 multiplications and 7 additions/subtractions.

The required LUT size for *n*-term Taylor-series expansion and *N* regions is $(n + 1) \times N$ words, and its MSB and 2nd MSB addresses αβ for

M=1.αβ ... are used for the corresponding data read. Table 7.18 shows the case for $n = 4$ and $N = 4$, and the LUT size is 20 words. Similarly, the optimal domain division can also be expressed in the format of Table 7.18.

Notice that as the number of the required terms for Taylor-series expansion for a specified accuracy is decreased thanks to our region division method, and then the number of LUT accesses also decreases.

Table 7.17 Required numbers of multiplications and additions/subtractions for *n*-term Taylor-series expansion.

| # of Taylor-series expansion terms(*n*) | # of multiplications | # of additions/ subtractions |
|---|---|---|
| 3 | 3 | 3 |
| 4 | 4 | 4 |
| 5 | 5 | 5 |
| 6 | 6 | 6 |
| 7 | 7 | 7 |
| 8 | 8 | 8 |

Table 7.18 LUT memory for 4 regions

| Address ($\alpha\beta$ $***$) | LUT data |
|---|---|
| $00***$ | $\alpha_0, \alpha_1, \alpha_2, \alpha_3$ , $\alpha_4$ for $a$ = 1.125 |
| $01***$ | $\alpha_0, \alpha_1, \alpha_2, \alpha_3$ , $\alpha_4$ for $a$ = 1.357 |
| $10***$ | $\alpha_0, \alpha_1, \alpha_2, \alpha_3$ , $\alpha_4$ for $a$ = 1.625 |
| $11***$ | $\alpha_0, \alpha_1, \alpha_2, \alpha_3$ , $\alpha_4$ for $a$ = 1.875 |

## 7.12 Summary

We have studied floating-point logarithmic algorithms with Taylor-series expansion with the mantissa region division uniformly and non-uniformly, and also with and without the mantissa region conversion. Further, we have investigated the logarithmic calculation to meet the specified accuracy with 2 terms of its Taylor-series expansion, for high-speed calculation. We have shown their hardware implementation trade-offs among simulation accuracy, numbers of multiplications/additions/subtractions and LUT sizes to meet various digital division specifications flexibly. The designer can build his/her conceptual dedicated hardware architecture design for logarithmic calculation with the contents described in this paper. The mantissa conversion with the uniform region division would be a reasonable choice in many cases.

We conclude this chapter by emphasizing that the previous works such as [23] have used the Taylor-series expansion for the logarithm calculation without region division. Notice that the region division methods are described in [34], but they are not for Taylor-series expansion. We here describe the mantissa region division method for Taylor-series expansion specifically which can reduce the number of calculations as the number of the division increases. Also this region division algorithm is applicable to the minimax polynomial approximation method described in [35].

# References

[1] J. Le Maire, N. Brunie, F. De Dinechin and J. Muller, "Computing Floating-point Logarithms with Fixed-point Operations", IEEE 23nd Symposium on Computer Arithmetic (ARITH), Silicon Valley, CA (Jul 2016).

[2] R. Prasad, S. Das, Kevin J. M. Martin, P. Coussy, “Floating-point CGRA Based Ultra-Low Power DSP Accelerator”, Journal of Signal Processing Systems, vol. 93, pp. 1159-1171 (Jan 2021).

[3] V. A. Lakshmi, G. F. Sudha, “A Novel Power Efficient 0.64-GFlops Fused 32-bit Reversible Floating-point Arithmetic Unit Architecture for Digital Signal Processing Applications”, Microprocessors and Microsystems, vol. 51, pp. 366–385 (Jun. 2017).

[4] J. R. D. Kumar, C. G. Babu, V. R. Balaji, C. Visvesvaran, “Analysis of Effectiveness of Power on Refined Numerical Models of Floating-point Arithmetic Unit for Biomedical Applications”, IOP Conference Series: Materials Science and Engineering (Mar. 2020).

[5] S. H. Farghaly, S. M. Ismail, “Floating-point Discrete Wavelet Transform-based Image Compression on FPGA”, International Journal of Electronics and Communications, vol. 124, pp. 1-11 (Jul. 2020).

[6] R. Omidi, S. Sharifzadeh, “Design of Low Power Approximate Floating-point Adders”, International Journal of Circuit Theory and Applications, vol. 49, no. 1, pp. 185-195 (Jun 2020).

[7] X. Ye, X. Tan, M. Wu, Y. Feng, D. Wang, H. Zhang, S. Pei, D. Fan, “An Efficient Dataflow Accelerator for Scientific Applications”, Future Generation Computer Systems, vol. 112, pp. 580-588 (Nov. 2020).

[8] M. M. Babu, K. R. Naidu, “Area and Power Efficient Fused Floating-point Dot Product Unit Based On Radix-2r Multiplier & Pipeline Feedforward-Cutset-Free Carry-Lookahead Adder”, Information Technology in Industry, vol. 9, no. 2, pp. 782-788 (Apr 2021).

[9] R. Murillo, A. A. D. Barrio, G. Botella, M. S. Kim, H. Kim, N. Bagherzadeh, “PLAM: A Posit Logarithm-Approximate Multiplier for Power Efficient Posit-based DNNs”, arXiv Computer Science, Machine Learning (Feb 2021).

[10] D. De Caro, M. Genovese, E. Napoli, N. Petra, A. G. M. Strollo, “Accurate Fixed-point Logarithmic Converter”, IEEE Transactions on Circuits and Systems II: Express Briefs, vol. 61, no. 7, pp. 526-530 (Jul. 2014).

[11] Y.-C. Liu, Y.-T. Chiang, T.-S. Hsu, C.-J. Liau, D.-W. Wa, “Floating-point Arithmetic Protocols for Constructing Secure Data Analysis Application”, Procedia Computer Science 22, Elsevier, vol. 22, pp. 152-161 (Oct. 2013).

[12] T. M. John, S. Chacko, “FPGA‐Based Implementation of Floating-point Processing Element for The Design Of Efficient FIR Filters”, IET Computers & Digital Techniques, pp. 296-301 (Mar. 2021).

[13] X. Lian, Z. Liu, Z. Song, J. Dai, W. Zhou, X. Ji, “High-Performance FPGA-based CNN Accelerator with Block-Floating-Point Arithmetic”, IEEE Transaction on Very Large Scale Integration Systems, vol. 27, no. 8, pp. 1874-1885 (Aug. 2019).

[14] T. Hoang, D. Le, C. Pham, “VLSI Design of Floating-point Twiddle Factor Using Adaptive CORDIC On Various Iteration Limitations”, IEEE 12th International Symposium on Embedded Multicore/Many-core Systems-on-Chip, Hanoi, Vietnam (Sep 2018).

[15] R. Pilipović, P. Bulić, U. Lotrič, "A Two-Stage Operand Trimming Approximate Logarithmic Multiplier”, IEEE Transaction on Circuits and Systems I: Regular Papers, vol. 68, No6, pp. 2535-2545 (Apr. 2021).

[16] C. Subhasri, B. R. Jammu, L. Harsha, N. Bodasingi, V. R. Samoju, “Hardware-Efficient Approximate Logarithmic Division with Improved Accuracy”, International Journal of Circuit Theory and Applications, vol. 49, no 1, pp. 128-141 (Nov. 2020).

[17] P. Drahoš, M. Kocúr, O. Haffner, E. Kučera, A. Kozáková, “RISC Conversions for LNS Arithmetic in Embedded Systems”, MDPI Mathematics, vol. 8, no. 8, pp. 1-17 (Jun. 2020).

[18] J. N. Mitchell Jr., “Computer Multiplication and Division using Binary Logarithms”, IRE Transaction on Electronic Computers, vol. EC-11, no. 4, pp. 512-517 (Aug. 1962).

[19] M. G. Arnold, T.A. Bailey, J. R. Cowles, J. J. Cupal, “Redundant Logarithmic Arithmetic”, IEEE Transaction on Computers, vol. 39, no. 8, pp. 1077-1086 (Aug. 1990).

[20] W. Liu, J. Xu, D. Wang, C. Wang, P. Montsuschi, “Design and Evaluation of Approximate Logarithmic Multipliers for Low Power Error-Tolerant Applications”, IEEE Transaction on Circuits and Systems I: Regular Paper, vol. 65, no. 9, pp. 2856-2868 (Sep 2018).

[21] R. C. Ismail, S. A. Z. Murad, R. Hussin, J. N. Coleman, “Improved Subtraction Function for Logarithmic Number System”, Procedia Engineering, vol. 53, pp. 387-392 (Dec. 2013).

[22] V. Paliouras, T. Stouraitis, “Low-Power Properties of the Logarithmic Number System”, IEEE Symposium on Computer Arithmetic. ARITH-15 2001, Vail, CO, USA (Jun. 2001).

[23] M. Mansour, A. M. El-Sawy, M. S. Aziz, A. T. Sayed. “A New Hardware Implementation of Base 2 Logarithm for FPGA”, International Journal of Signal Processing Systems, vol. 3, no. 2, pp. 177-182 (Dec. 2015).

[24] Y. Luo, Y. Wang, Y. Ha, Z. Wang, S. Chen, H. Pan, “Generalized Hyperbolic CORDIC and its Logarithmic and Exponential Computation with Arbitrary Fixed Base”, IEEE Transaction on Very Large Scale Integration Systems, vol. 27, no. 9, pp. 2156-2169 (Sep. 2019).

[25] J. Wei, A. Kuwana, H. Kobayashi, K. Kubo, Y. Tanaka, "Floating-point Inverse Square Root Algorithm based on Taylor-series Expansion", IEEE Transaction on Circuits and Systems II: Express Briefs, vol. 64, no. 7, pp. 2640-2644 (Feb. 2021).

[26] J. Wei, A. Kuwana, H. Kobayashi, K. Kubo, “Revisit to Floating-point Division Algorithm based on Taylor-series Expansion”, IEEE Asia Pacific Conference on

Circuits and Systems, Ha Long, Vietnam (Dec. 2020).

[27] Y. Chatelain, E. Petit, P. O. Castro, G. Lartigue, D. Defour, “Automatic Exploration of Reduced Floating-point Representations in Iterative Methods”, In: Yahyapour R. (eds) Euro-Par 2019: Parallel Processing. Lecture Notes in Computer Science, Springer, Cham, vol. 11725, pp. 481-494 (Aug. 2019).

[28] K. Isupov, “High-Performance Computation in Residue Number System using Floating-point Arithmetic”, MDPI Computation, vol. 9, no. 2, 9, pp. 55-62 (Jan, 2021).

[29] S. Vollala, B. S. Begum and N. Ramasubramanian, “Hardware Design for Multiplicative Modular Inverse based on Table Look Up Technique”, International Conference on Computing and Network Communications, Trivandrum, India (Dec. 2015).

[30] S. Hsiao, C. Wen, H. Lee, “Implementation of Floating-point CORDIC Rotation and Vectoring Based on Look Up Tables and Multipliers”, International Symposium on Next Generation Electronics, Kaohsiung, Taiwan (Nov. 2010).

[31] R. D. Sharma, Y. Agrawal, R. Parekh, “Design of Prominent Single-Precision 32-Bit Floating-point Adder Using Single-Electron Transistor Operating At Room Temperature”, In: Patel Z., Gupta S., Kumar Y. B. N. (eds.) Advances in VLSI and Embedded Systems. Lecture Notes in Electrical Engineering, Springer, Singapore, vol. 676, pp. 201-209 (Aug. 2020).

[32] S. Sushma1, S. K. Ravindran1, P. R. Nadagoudar1, P. A. Sophy, “Implementation of a 32-bit RISC Processor with Floating-point Unit in FPGA Platform”, Journal of Physics: Conference Series, Vol. 1716, No 012047, pp.1-9 (Dec. 2020).

[33] E. Ali, W. Pora, “Implementation and Verification of IEEE-754 64-bit Floating-point Arithmetic Library for 8-bit Soft-Core Processors”, 8th International Electrical Engineering Congress, Chiang Mai, Thailand (Mar. 2020).

[34] T.-B. Juang, S.-H. Chen, H.-J. Cheng, “A Lower Error and ROM-Free Logarithmic Converter for Digital Signal Processing Applications”, IEEE Transaction on Circuits

and Systems – II, vol. 56, no. 12, pp. 931-935 (Dec. 2009).

[35] N. Brisebarre, S. Chevillard, “Efficient Polynomial $L^{\infty}$-Approximations”, IEEE Symposium on Computer Arithmetic, Montpellier, France (Jun. 2007).

# Chapter 8

# CONCLUSIONS AND FUTURE WORK

## 8.1 Conclusion

The floating-point arithmetic units are widely used in today's DSPs, CPUs and GPUs, and some graphics chips have thousands of floating-point arithmetic units integrated into them, as they can represent a larger range of numbers with greater precision than integers. At the same time, with advances in semiconductor technology, various floating-point arithmetic units with complex arithmetic functions are gradually appearing in some special application chips. Therefore, VLSI implementation of high-performance floating-point arithmetic units has been a hot research topic in both academia and industry.

In this part, the computation of arithmetic algorithms using Taylor-series expansions is investigated. The innovative result of the dissertation is the use of the “divide and conquer” technique in the floating-point range using Taylor-series expansion. Three methods are proposed for fast calculation with high efficiency: (i) mantissa region uniform division, (ii) mantissa region non-uniform division, (iii) mantissa region conversion. In this part, the computation of arithmetic algorithms using Taylor-series expansions is investigated. In the part, the algorithms for floating-point division, square root, inverse square root, exponential and Logarithmic arithmetic units are analyzed in depth. We also elucidate trade-offs between

LUT size, arithmetic precision and usage of basic arithmetic operations: addition, subtraction and multiplication. A designer can build efficient hardware devices by the method proposed in this paper. Hardware design as implemented depends on the design criteria and available hardware elements.

## 8.2 Future Work

In subsequent work, the proposed algorithms are implemented in FPGAs and the power consumption, delay, and chip area of the circuit using the proposed algorithms applied in hardware are discussed.

As IC process technology advances, future work should also discuss how to improve computational accuracy, reduce delay, power consumption and reduce VLSI area using algorithms such as the Newton-Raphson algorithm, the Goldschmidt algorithm and the digital iterative algorithm.

# List of Publications

## Journal Paper

[1] **Jianglin WEI,** Anna Kuwana, Haruo Kobayashi, Kazuyoshi Kubo, "IEEE754 Binary32 Floating-Point Logarithmic Algorithms based on Taylor-Series Expansion with Mantissa Region Conversion and Division", IEICE TRANSACTIONS on Fundamentals of Electronics, Communications and Computer Sciences, Vol.E105-A, No.7, pp.-, Jul. 2022.

[2] **Jianglin Wei**, Anna Kuwana, Haruo Kobayashi, Kazuyoshi Kubo, Yuuki Tanaka, "Floating-point Inverse Square Root Algorithm Based on Taylor-series Expansion", IEEE Transactions on Circuits and Systems II: Express Briefs, Vol. 68, Issue 7, pp. 2640-2644, (Jul. 2021).

[3] **Jianglin Wei**, Nene Kushita, Keno Sato, Takashi Ishida, Toshiyuki Okamoto, Tamotsu Ichikawa, Hirotaka Arai, Lei Sha, Anna Kuwana, Takayuki Nakatani, Kazumi Hatayama, Haruo Kobayashi, "Short-Time INL Testing Methodology for High-Resolution ΔΣ ADC", Journal of Mechanical and Electrical Intelligent System (JMEIS, J. Mech. Elect. Intel. Syst.). vol. 3, no. 2, pp.87-101 (May. 2020).

## Review Paper

[1] 小林春夫、桑名杏奈、**魏 江林**、築地伸和、趙 宇杰.「IoT 時代のアナログ/ミクストシグナル回路テスト技術」電気学会論文誌 (論文誌 C), vol. 141, no. 1, pp.1-12 (2021 年 1 月).

## International Conference


[1] **Jiang-Lin Wei,** "ΔΣ ADC Linearity Testing Technology and Floating-point Arithmetic Algorithms with Taylor-series Expansion", ATS Doctoral Thesis Award Contest, 30th IEEE Asian Test Symposium (ATS 2021), Virtual Event Hosted by Japan (Nov. 2021).

[2] **Jianglin Wei**, Anna Kuwana, Haruo Kobayashi, Kazuyoshi Kubo, "Divide and Conquer: Floating-point Exponential Calculation Based on Taylor-series Expansion", IEEE 14th International Conference on ASIC (ASICON 2021), Virtual Event, Kunming, China (Oct. 2021).

[3] **Jianglin Wei,** Anna Kuwana, Haruo Kobayashi, Kazuyoshi Kubo, Yuuki Tanaka, "Examination of Optimal Domain Division in Floating-point Arithmetic Using Taylor-series Expansion", 30th International Workshop on Post-Binary ULSI Systems, Fully Virtual (May. 2021)

[4] **Jianglin Wei,** Anna Kuwana, Haruo Kobayashi, Kazuyoshi Kubo, "Floating-point Square Root Calculation Algorithm Based on Taylor-series Expansion and Region Division, IEEE 64th International Midwest Symposium on Circuits and Systems (MWSCAS2021), Virtual Event, East Lansing, Michigan, USA. (Ang. 2021)

[5] **Jianglin Wei,** Anna Kuwana, Haruo Kobayashi and Kazuyoshi Kubo, "Revisit to Floating-point Division Algorithm Based on Taylor-series Expansion", The 16th IEEE Asia Pacific Conference on Circuits and Systems (APCCAS), Virtual Event, Ha Long Bay, Vietnam, (Dec. 2020).

[6] **Jianglin Wei,** Nene Kushita, Takahiro Arai, Lei Sha, Anna Kuwana, Haruo Kobayashi, Takayuki Nakatani, Kazumi Hatayama, Keno Sato, Takashi Ishida, Toshiyuki Okamoto, Tamotsu Ichikawa, "High-Resolution Low-Sampling-Rate ΔΣADC Linearity Short-Time Testing Algorithm", 2019 13th IEEE International Conference on ASIC (ASICON 2019), Chongqing, China (Oct. 2019).

[7] **Jianglin Wei**, Nene Kushita, Takahiro Arai, Lei Sha, Anna Kuwana, Haruo Kobayashi, Takayuki Nakatani, Kazumi Hatayama, Keno Sato, Takashi Ishida, Toshiyuki Okamoto, Tamotsu Ichikawa, “High-Resolution Low-Sampling-Rate ΔΣ ADC Linearity Testing Algorithm”, 3rd International Conference on Technology and Social Science (ICTSS2019), Kiryu, Japan (May. 2019).

[8] **Jianglin Wei,** Nene Kushita, Takahiro Arai, Lei Sha, Anna Kuwana, Haruo Kobayashi, Takayuki Nakatani, Kazumi Hatayama (Gunma Univ.), Keno Sato, Takashi Ishida, Toshiyuki Okamoto, Tamotsu Ichikawa (ROHM Semiconductor Co., Ltd.), “Algorithm for ΔΣADC Linearity Test in Short Time”, 5th Taiwan and Japan Conference on Circuits and Systems (TJCAS 2019 at Nikko), Nikko, Tochigi, Japan, (Aug. 2019).

[9] **Jiang-Lin Wei,** Nene Kushita and Haruo Kobayashi, "Limit Cycle Manage Using Random Signal in ΔΣ DA Modulator", 5th International Symposium of Gunma University Medical Innovation and 9th International Conference on Advanced Micro-Device Engineering, Kiryu, Japan, (Dec. 2018).

[10] **Jianglin Wei,** Nene Kushita, Haruo Kobayashi, “Limit Cycle Suppression Technique Using Random Signal In ΔΣ DA Modulator”, IEEE 14th International Conference on Solid-State and Integrated Circuit Technology, Qingdao, China, (Nov. 2018).

[11] Keno Sato, Takayuki Nakatani, Takashi Ishida, Toshiyuki Okamoto, Tamotsu Ichikawa, Shogo Katayama, Gaku Ogihara, Daisuke Iimori, Yujie Zhao, **Jianglin Wei**, Anna Kuwana, Kazumi Hatayama, Haruo Kobayashi, “High Precision Measurement of Sub-Nano Ampere Current in ATE Environment", 30th IEEE Asian Test Symposium (ATS 2021), Virtual Event Hosted by Japan (Nov. 2021).

[12] Gaku Ogihara, Takayuki Nakatani, Daisuke Iimori, Shogo Katayama, Anna Kuwana, Keno Sato, Takashi Ishida, Toshiyuki Okamoto, Tamotsu Ichikawa, Yujie Zhao, **Jianglin Wei,** Kazumi Hatayama, Haruo Kobayashi, "Evaluation of High-Precision Nano-Ampere Current Measurement Method for Mass

Production", 28th IEEE International Conference on Electronics Circuits and Systems (IEEE ICECS 2021), Virtual Event, Dubai, UAE, (Nov, 2021).

[13] Daisuke Iimori, Takayuki Nakatani, Shogo Katayama, Gaku Ogihara, Akemi Hatta, Anna Kuwana, Keno Sato, Takashi Ishida, Toshiyuki Okamoto, Tamotsu Ichikawa, **Jianglin Wei,** Yujie Zhao, Tri Minh Tran, Kazumi Hatayama, Haruo Kobayashi, "Summing Node and False Summing Node Methods: Accurate Operational Amplifier AC Characteristics Testing without Audio Analyzer", 51st IEEE International Test Conference (ITC 2021), Virtual Event, (Oct. 2021).

[14] Keno Sato, Takashi Ishida, Toshiyuki Okamoto, Tamotsu Ichikawa, **Jianglin Wei**, Takayuki Nakatani, Yujie Zhao, Shogo Katayama, Shuhei Yamamoto, Anna Kuwana, Kazumi Hatayama, Haruo Kobayashi, "Revisit to Accurate ADC Testing with Incoherent Sampling Using Proper Sinusoidal Signal and Sampling Frequencies", 51st IEEE International Test Conference (ITC 2021), Virtual Event, (Oct. 2021).

[15] Xiongyan Li, Tianrui Feng, Lengkhang Nengvang, Shogo Katayama, **Jianglin Wei,** Haijun Lin, Kazufumi Naganuma, Kiyoshi Sasai, Junichi Saito, Anna Kuwana, Haruo Kobayashi, "Folding ADC for Multi-bit ΔΣ AD Modulator", International Conference on Analog VLSI Circuits (AVIC 2021), Virtual Event, Bordeaux, France (Oct. 2021).

[16] Yujie Zhao, Anna Kuwana, Shogo Katayama, **Jianglin Wei,** Haruo Kobayashi, Takayuki Nakatani, Kazumi Hatayama, Keno Sato, Takashi Ishida, Toshiyuki Okamoto, Tamotsu Ichikawa, "Code Selective Histogram Method: Two-Tone Signal for ADC Linearity Test Time Reduction", International Conference on Analog VLSI Circuits (AVIC 2021), Virtual Event, Bordeaux, France (Oct. 2021).

[17] Lengkhang Nengvang, Shogo Katayama, **Jianglin Wei,** Lei Sha, Tri Minh Tran, Anna Kuwana, Kazufumi Naganuma, Kiyoshi Sasai, Junichi Saito, Haruo Kobayashi, "Two-Step Incremental ADC Architecture With Self-Calibration of

Two Reference Voltages Ratio", International Conference on Analog VLSI Circuits (AVIC 2021), Virtual Event, Bordeaux, France (Oct. 2021).

[18] Haruo Kobayashi, Xueyan Bai, Yujie Zhao, Shuhei Yamamoto, Dan Yao, Manato Hirai, J**ianglin Wei**, Shogo Katayama, Anna Kuwana, "Classical Mathematics and Analog/Mixed-Signal IC Design", IEEE 14th International Conference on ASIC (ASICON 2021), Virtual Event, Kunming, China (Oct. 2021).

[19] Shuhei Yamamoto, Yuto Sasaki, Yujie Zhao, **Jianglin Wei,** Anna Kuwana, Keno Sato, Takashi Ishida, Toshiyuki Okamoto, Tamotsu Ichikawa, Takayuki Nakatani, Tri Tran, Shogo Katayama, Kazumi Hatayama, Haruo Kobayashi, "Metallic Ratio Equivalent-Time Sampling: A Highly Efficient Waveform Acquisition Method", the 27th IEEE International Symposium on On-Line Testing and Robust System Design (IOLTS'21), Virtual event (Jun. 2021).

[20] Yujie Zhao, Anna Kuwana, Shuhei Yamamoto, Yuto Sasaki, Haruo Kobayashi, Tri Minh Tran, Takayuki Nakatani, Kazumi Hatayama, Keno Sato, Takashi Ishida, Toshiyuki Okamoto, Tamotsu Ichikawa, **Jianglin Wei**, Shogo Katayama, "Input Signal and Sampling Frequencies Requirements for Efficient ADC Testing with Histogram Method", The 36th International Technical Conference on Circuits/Systems, Computers and Communications (ITC-CSCC 2021), Virtual Event, (Jun. 2021)

[21] Riho Aoki, **Jianglin Wei,** Yujie Zhao, Anna Kuwana, Haruo Kobayashi, Takayuki Nakatani, Kazumi Hatayama, Keno Sato, Takashi Ishida, Toshiyuki Okamoto, Tamotsu Ichikawa, "Analysis, Testing and Calibration of Charge Distribution SAR ADC Architecture with Split Capacitor", 4th International Conference on Technology and Social Science (ICTSS 2020), Virtual Event, Kiryu, Japan, (Dec. 2020).

[22] (Invited) Haruo Kobayashi, Anna Kuwana, **Jianglin Wei,** Yujie Zhao, Shogo Katayama, Tran Minh Tri, Manato Hirai, Takayuki Nakatani, Kazumi Hatayama, Keno Sato, Takashi Ishida, Toshiyuki Okamoto, Tamotsu

Ichikawa, “Analog/Mixed-Signal Circuit Testing Technologies in IoT Era”, IEEE 15th International Conference on Solid-State and Integrated Circuit Technology, Virtual Event, Kunming, China (Nov. 2020).

[23] Gaku Ogihara, Takayuki Nakatani, Akemi Hatta, Keno Sato, TakashiIshida, Toshiyuki Okamoto, Tamotsu Ichikawa, Anna Kuwana, Riho Aoki, Shogo Katayama, **Jianglin Wei,** Yujie Zhao, JianlongWang, Kazumi Hatayama, Haruo Kobayashi, “Summing Node Test Method: Simultaneous Multiple AC Characteristics Testing of Multiple Operational Amplifiers”, 29th IEEE Asian Test Symposium, Virtual Event, Penang, Malaysia, (Nov. 2020).

[24] Keno Sato, Takashi Ishida, Toshiyuki Okamoto, Tamotsu Ichikawa (Rohm Semiconductor), **Jiang-Lin Wei,** Nene Kushita, Hirotaka Arai, Lei Sha, Anna Kuwana, Takayuki Nakatani, Kazumi Hatayama, Haruo Kobayashi (Gunma University), "An Effective INL Test Methodology For Low Sampling Rate and High Resolution Analog-to-Digital Converter", IEEE International Test Conference, Washington, D. C. (Nov. 2019).

[25] Keno Sato, Takashi Ishida, Toshiyuki Okamoto, Tamotsu Ichikawa (Rohm Semiconductor), **Jiang-Lin Wei**, Nene Kushita, Hirotaka Arai, Anna Kuwana, Takayuki Nakatani, Kazumi Hatayama, Haruo Kobayashi (Gunma University) "An FFT-based INL Prediction Methodology for Low Sampling Rate and High Resolution Analog-to-Digital Converter", IEEE VLSI Test Symposium, Monterey, CA (Apr. 2019).

[26] Haruo Kobayashi, **Jianglin Wei**, Masahiro Murakami, Jun-ya Kojima, Nene Kushita, Yuanyang Du, Jianlong Wang, “Performance Improvement of ΔΣ ADC/DAC/TDC Using Digital Technique”, IEEE 14th International Conference on Solid-State and Integrated Circuit Technology, Qingdao, China (Nov. 2018).

**Domestic Conferences / Seminars**

[1] **魏 江林**, 桑名杏奈, 小林春夫（群馬大学）, 久保和良（小山高専）, 田中勇樹（群馬大学）. 「テイラー展開を用いた浮動小数点演算での最適領域分割の検討」2020 年度（第 11 回）電気学会東京支部栃木・群馬支所合同研究発表会, オンライン開催（2021 年 3 月).

[2] **魏 江林**, 桑名杏奈, 小林春夫, 久保和良, 田中勇樹, 「テイラー展開を用いたいくつかの 2 進浮動小数点演算アルゴリズムの検討」 第 34 回多値論理とその応用研究会, オンライン開催（2021 年 1 月).

[3] **魏 江林,** 桑名杏奈, 小林春夫, 久保和良 .「テイラー展開を用いたデジタル除算アルゴリズム」第 43 回 多値論理フォーラム, WEB 開催(2020 年 9 月).

[4] **魏 江林**, 串田 弥音, 桑名 杏奈, 沙 磊, 小林 春夫, 中谷 隆之, 畠山 一実, 佐藤賢央, 石田 嵩, 岡本 智之, 市川 保. 「高分解能低速 ΔΣADC 線形性の量産試験アルゴリズム —5 次非線形性の場合—」2019 年度 第 10 回 電気学会栃木・群馬支所合同研究発表会, 群馬高専（2020 年 3 月).

[5] **魏 江林**, 串田 弥音, 新井 宏崇, 桑名 杏奈, 沙 磊, 小林 春夫, 中谷 隆之, 畠山 一実, 佐藤 賢央, 石田 嵩, 岡本 智之, 市川 保.「FFT 法

を用いたΔΣADC 線形性試験アルゴリズム」平成 30 年度　第 9 回 電気学会東京支部栃木・群馬支所 合同研究発表会，小山高専 （2019 年 3 月）.

[6] **魏 江林**,「ΔΣDA 変調器でのランダム信号を用いたリミットサイクル抑制技術」第 71 回システム LSI 合同ゼミ 、埼玉大学,（2019 年 1 月）.

[7] 荻 原岳，荻原岳，中谷隆之，片山翔吾，飯森大翼，八田朱美，桑名杏奈（群馬大学），佐藤賢央，石田嵩，岡本智之，市川保（ローム（株）），**魏 江林,** 趙 宇杰，チャン・ミン・チー，畠山一実，小林春夫（群馬大学)「複数オペアンプ複数 AC 特性の並列試験技術サミングノード法の検討」第 83 回 FTC 研究会，オンライン（2021 年 7 月）.

[8] 趙 宇杰，桑名 杏奈，山本修平，佐々木 優斗，小林 春夫，チャン ミン チー，片山翔吾，**魏 江林**，中谷 隆之，畠山 一実（群馬大学）佐藤賢央，石田 嵩，岡本 智之，市川 保 （ローム（株））．ヒストグラム法による効率的 ADC 試験のための入力周波数とサンプリング周波数の関係の検討 第 83 回 FTC 研究会，オンライン（2021 年 7 月）.

[9] ネンワンレーンカン，**魏 江林**，片山翔吾，沙磊，桑名杏奈（群馬大学），永沼和文，篠井潔，斉藤潤一（アルプスアルパイン(株)），小林春夫（群馬大学）．「バンドパスΔΣAD 変調器への FIR DAC 適用の検討」, 2020

年度（第 11 回）電気学会東京支部栃木・群馬支所合同研究発表会，オンライン開催（2021 年 3 月).

## Award

[1] **Jianglin Wei**, Nene Kushita, Takahiro Arai, Lei Sha, Anna Kuwana, Haruo Kobayashi, Takayuki Nakatani, Kazumi Hatayama, Keno Sato, Takashi Ishida, Toshiyuki Okamoto, Tamotsu Ichikawa, “High-Resolution Low-Sampling-Rate ΔΣ ADC Linearity Testing Algorithm”, 3rd International Conference on Technology and Social Science (ICTSS2019), Kiryu, Japan (May. 2019). **(BEST STUDENT PAPER AWARD)**